VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Výkonové testování intrusního detekčního systému s využitím technologie CUDA

Performance Testing of Intrusion Detection System Using CUDA Technology

2016 **Radek Mandrla**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Radek Mandrla**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Výkonové testování intrusního detekčního systému s využitím technologie CUDA
Performance Testing of Intrusion Detection System Using CUDA Technology

Jazyk vypracování: čeština

Zásady pro vypracování:

Technologie CUDA v současnosti představuje prostředek pro urychlení náročných výpočetních úloh, jakou může být například monitorování a analýza síťového provozu pomocí intrusních detekčních či protekčních systémů (IDS, IPS). Jedním z nejrozšířenějších open-source IDS současnosti je nástroj Suricata, která právě technologii CUDA podporuje. Cílem diplomové práce je zpracovat detailní výkonové testování IDS Suricata v testovací topologii s IP telefonní infrastrukturou a porovnat výkonové statistiky pro provoz s/bez technologie CUDA.

1. Popište technologie CUDA, IDS Suricata, bezpečnostní rizika v IP telefonii.
2. Potvrďte detailní přehled nástrojů pro penetraci v IP telefonii.
3. Vytvořte testovací topologii a proveďte konfiguraci IDS Suricata.
4. Realizujte měření výkonostních charakteristik IDS s využití a bez využití CUDA.
5. Vyhodnoďte výkonostní testy s cílem definovat optimální konfiguraci nasazení IDS v síťovém provozu.

Seznam doporučené odborné literatury:

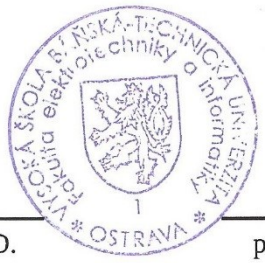
[1] Chris Sanders, Jason Smith: Applied Network Security Monitoring: Collection, Detection, and Analysis, 2013.
[2] Jason Sanders, Jason Sanders: CUDA by Example: An Introduction to General-Purpose GPU Programming, 2010.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Filip Řezáč**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016



doc. Ing. Miroslav Vozňák, Ph.D.	prof. RNDr. Václav Snášel, CSc.
vedoucí katedry	*děkan fakulty*

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *9. dubna 2016*

.....
podpis studenta

Poděkování

Velmi rád bych poděkoval svému vedoucímu diplomové práce Ing. Filipu Řezáčovi, Ph.D. za pomoc při výběru tématu, sestavení obsahu a připomínky, které mi byly cenným vodítkem při vypracování této práce. Také bych chtěl poděkovat své manželce a dětem za trpělivost.

Abstrakt

Tato diplomová práce se zabývá výkonovým testováním intrusivního detekčního systému Suricata nad technologií CUDA. V teoretické části jsou uvedeny výkonové možnosti technologií využívajících ke své práci grafické procesory, v našem případě od firmy NVIDIA. Tyto možnosti jsou porovnávány s výpočetními možnostmi procesoru počítače. Výhody použití technologie CUDA jsou pak prezentovány v jedné z praktických oblastí využití, kdy se jedná o zajištění bezpečnosti provozu počítačové sítě intrusivním detekčním nebo prevenčním systémem. Praktická část je poté zaměřena na samotné výkonové testování s cílem analyzovat výpočetní výkon technologie CUDA v porovnání s klasickým procesorem v prostředí intrusivních detekčních systémů. Výhody této technologie jsou prezentovány v rychlosti zpracování informace a dále snížením výpočetního výkonu procesoru počítače, který je přenesen na grafický procesor a jeho jádra. V závěru je na základě výsledků testování srovnáno, v kterých oblastech se přínos technologie CUDA nejvíce projevil.

Klíčová slova

CUDA, Suricata, IDS (Systém detekce průniku), VoIP, DoS, SIP

Abstract

The thesis deals with the performance testing of the Intrusion Detection System Suricata over CUDA technology. In the theoretical part, the performance possibilities of the technologies using graphic processors are introduced, in this case those from NVIDIA company. These possibilities are compared with the calculation capabilities of the computer processor. The advantages of using CUDA technology are then presented in one of the practical fields of application, such as ensuring a secure run of the computer network by either the intrusion detection or the prevention system. The practical part of the thesis then focuses on the performance testing itself, with a purpose to analyze the calculation performance of CUDA technology in comparison to the common processor in the environment of intrusion detection systems. The advantages of this technology are presented by the speed of information processing and also by the decrease of calculation performance of the computer processor, which is transfered to the graphic processor and its cores. In conclusion, according to the testing results, there is a comparison showing the areas where the benefit of CUDA technology is most significant.

Key words

CUDA, Suricata, IDS (Intrusion Detection System), VoIP, DoS, SIP

Obsah

Seznam použitých zkratek

ARP 802.1q	**Address Resolution Protocol**
CDP	**Cisco Discovery Protocol**
CPU	**Central Processing Unit**
CUDA	**Compute Unified Device Architecture**
DHCP	**Dynamic Host Configuration Protocol**
DIDS	**Distributed Intrusion Detection System**
DoS	**Denial of Service**
GCC	**GNU Compiler Collection**
GPU	**Graphic Processing Unit**
HIDS	**Host-based Intrusion Detection System**
IDS	**Intrusion Detection System**
ISO/OSI	**Open Systems Interconnection**
LLDP-MED	**Link Layer Discovery protocol - Media Endpoint Discovery**
MiTM	**Man in The Middle**
NIDS	**Network Intrusion Detection System**
PBX	**Private branch exchange**
QoS	**Quality of Service**
RTCP	**RTP Control protokol**
RTP	**Real-time Transport Protocol**
SDP	**Session Description Protocol**
SIP	**Session Initiation Protocol**
SLI	**Scalable Link Interface**
SRTCP	**Secure RTCP**
SRTP	**Secure RTP**
Suricata	**IDS aplikace**
TLS	**Transport Layer Security**
UA	**User Agent**
UAC	**User Agent Client**
UAS	**User Agent Server**
UDP	**Unified Datagram Protocol**
VoIP	**Voice over Internet Protocol**
XML	**Extensible Markup Language**
ZRTP	**Zimmermann RTP**

Seznam obrázků

Seznam tabulek

Úvod

Diplomová práce se zabývá výkonovým testováním intrusivního detekčního systému (IDS) Suricata využívající výpočetní výkon grafických jader procesorů pomocí technologie CUDA. Intrusivní detekční systémy jsou nedílnou součástí zabezpečení počítačových sítí, jelikož rozsáhlé síťové topologie je v reálném čase těžké testovat pravidelnou kontrolou na přítomnost nežádoucího chování některého z prvků sítě. V teoretické části je představen intrusivní detekční systém Suricata, technologie CUDA od firmy NVIDIA a praktické řešení IP telefonie, včetně stručného popisu teorie přenosu hlasové informace s využitím technologie VoIP.

Ve třetí části jsou uvedena jednotlivá rizika VoIP telefonie na protokolu SIP a je zde uveden podrobný výčet nástrojů sloužících k testování zabezpečení. Některé nástroje budou použity v praktické části, zejména ty, které výkonnostně zatíží hardware testovaného sestavy a umožní objektivně porovnat výsledek testování ve vztahu k zatížení procesoru počítače, grafického procesoru grafické karty podporující technologii CUDA a rychlosti zpracování paketů v dané síťové topologii.

Ve čtvrté části je uveden postup praktické instalace jednotlivých komponent testovací topologie, to znamená IDS/IPS Suricata, telefonní ústředna Asterisk a její konfigurace a nastavení SIP účtů. IDS/IPS Suricata byl nakonfigurován, aby využíval technologii CUDA a poté byla provedena konfigurace bez podpory této technologie, kdy byl porovnáván rozdíl výkonu IDS/IPS Suricata.

V páté části je uveden praktický postup konfigurace jednotlivých penetračních nástrojů za účelem vyvolat co největší výkonové zatížení testované sítě a dále nastavení pravidel IDS/IPS Suricata. Tato pravidla byla úmyslně nastavena tak, aby došlo k zachycení útoku testovacích nástrojů, ale nedošlo k přerušení datového toku ze zdroje provozu penetračního nástroje. Suricata byla tedy nastavena jako intrusivní detekční systém a ne jako intrusivní prevenční systém. Záměrem bylo porovnat, jak velkou výkonovou výhodu má konfigurace IDS/IPS Suricata s podporou CUDA oproti konfiguraci IDS/IPS Suricata bez podpory CUDA technologie.

V závěrečné části jsou výsledky z praktické části zhodnoceny, kde je porovnání výsledků různých opakovaných měření vyhodnoceno z pohledu zatížení procesoru počítače s využitím technologie CUDA oproti zatížení procesoru počítače bez podpory technologie CUDA a dále z pohledu počtu zachycených paketů a datového objemu IDS/IPS Suricata, který využívá ke své činnosti jádra grafického procesoru anebo pouze jádra procesoru počítače.

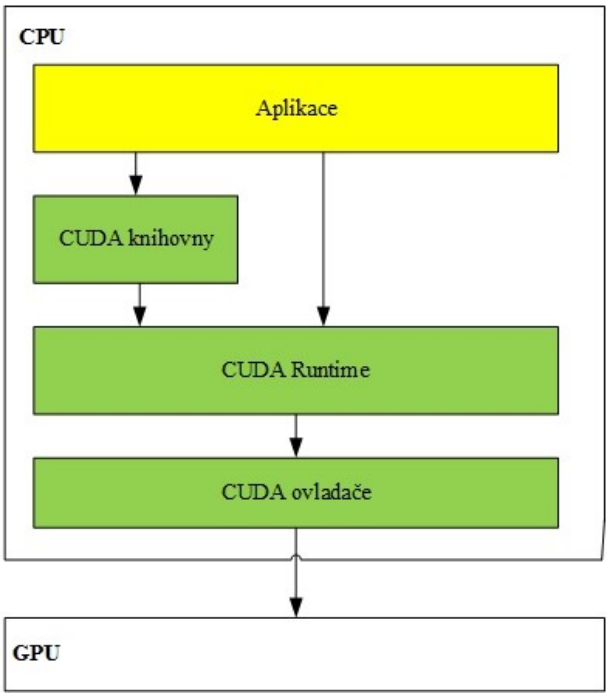
1 Teorie a úvod do technologií využitých v diplomové práci

V této kapitole bude uvedena technologie CUDA, na jakých principech pracuje a jak ji lze využít v masivních výpočetních operacích při zajištění bezpečnosti počítačové sítě, konkrétně intrusivním detekčním a prevenčním systémem Suricata. Dále bude uvedena teorie přenosu hlasové informace s využitím technologie VoIP (Voice over Internet Protocol), kde budou představeny transportní a signalizační protokoly, konkrétně protokol SIP.

1.1 Technologie CUDA

V této kapitole bude vysvětleno paralelní využití jader grafických procesorů a představena technologie CUDA od společnosti NVIDIA. Dále bude uvedeno, jak bude tato technologie využita ve vztahu k diplomové práci.

Tato technologie vznikla z potřeby zvýšit výpočetní výkon, kdy je efektivnější rozložit složitý výpočet do několika jednodušších výpočtů. Společnost NVIDIA proto začala pracovat na technologii, která výpočetní výkon rozloží do více vláken. Oficiálně byla představena pod názvem CUDA v roce 2006 a nyní (prosinec 2015) je ke stažení verze CUDA 7.5 [1], [2]. Jedná se tedy o hardwarovou a softwarovou architekturu (obr. 1.1).

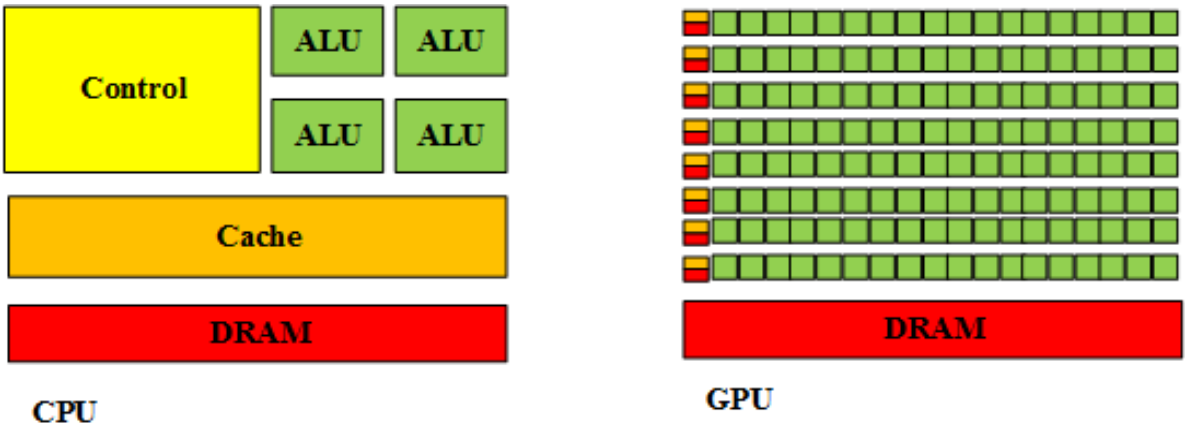


Obrázek 1.1 *Softwarová a hardwarová architektura.*

V současné době jsou v nabídce 12 jádrové procesory společností zabývajících se jejich vývojem, což umožňuje vykonávání instrukcí procesoru i na 96 vláknech [7]. Oproti tomu například grafická karta GeForce GTX TITAN Z má 5760 jader [3][8] a právě technologie CUDA dokáže využít výkon těchto

jader grafického procesoru (GPU). Nejlépe lze však výkon využít při zpracování operací, kde nejsou kladeny nároky na logiku, ale na zpracování velkého počtu paralelních instrukcí. Procesor počítače (CPU) je však stále řídícím prvkem, který "pouze" využívá výkon stream procesorů GPU. Instrukce SIMD umožňuje vykonávat jeden postup nad větším množství dat současně. Proces využití technologie CUDA je postaven tak, že jsou data z paměti RAM nakopírována do paměti grafické karty, kde se automaticky rozdělí výpočet mezi výpočetní jádra. Výpočet, který se má provést na jednotlivých jádrech grafického procesoru je instruován procesorem počítače, a po skončení výpočtu jsou výsledná data přenesena do paměti RAM.

Na ploše čipu grafického akcelerátoru je velké množství skalárních procesorů seskupených do streaming multiprocesorů. Jedná se o SIMT (Single-Instruction, Multiple-Thread) architekturu, která má malou vyrovnávací paměť [9], viz obrázek 1.2.



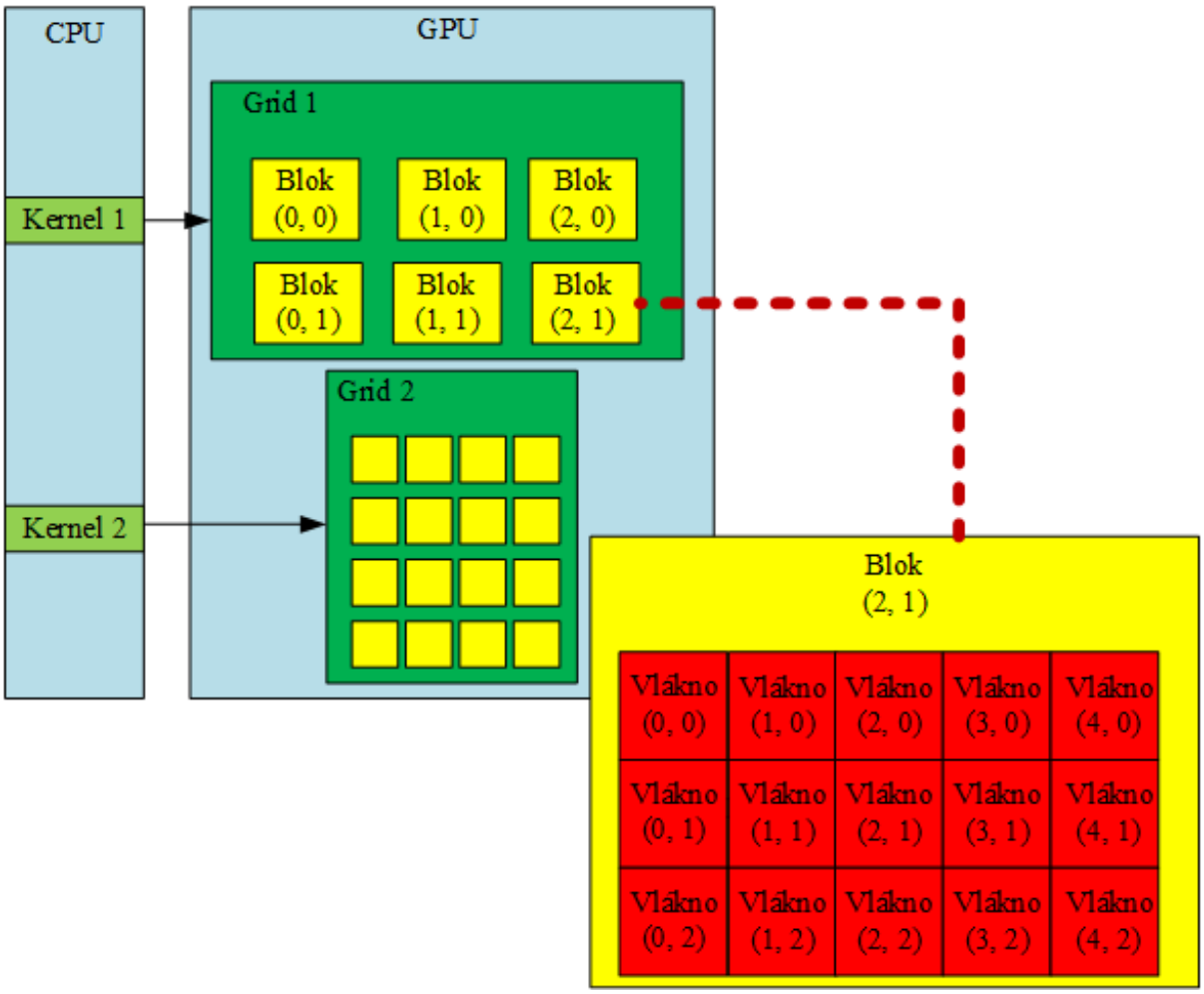
Obrázek 1.2 *Architektura CPU a GPU.*

CUDA poskytuje několik rozšíření jazyka C a C++. Programátor si může vybrat k realizaci jeho aplikace jazyk C, C++, Fortran nebo standard OpenCL nebo DirectX [4].

1.1.1 SIMD a SIMT

Při paralelním programování architektura SIMD provádí stejnou operaci na velkém počtu dat (například seznamy). Výpočet každého z prvků probíhá nezávisle na ostatních, což se využívá při zpracování zvuku a videa. Soubor procesorů řízený centrální jednotkou pracuje synchronně po instrukcích [10].

U architektury SIMT je výpočet rozčleněn do velkého množství vláken stejné struktury. Jednu instrukci zpracovává několik vláken. Balík vláken, který je zpracovávaný v jednom okamžiku se nazývá warp, jehož velikost závisí na počtu výpočetních jednotek. Kolekce datových elementů stejného typu, které vyžadují stejné zacházení, se nazývá stream neboli datový proud. Kernel je skupina funkcí aplikovaných na jednotlivé elementy datového proudu. Na každý datový proud může být aplikováno několik kernelů, viz obrázek 1.3.



Obrázek 1.3 *Kernel, Grid, Blok, Vlákno.*

Architektura SIMT využívá lokální paměť na čipu. V rámci jednoho kernelu je aplikována jedna funkce na jeden datový element. Výpočet nad všemi elementy může probíhat paralelně, bez čekání na výsledek operace na předchozí element. Data jsou načtena, zpracována a zaslána na výstup. Tyto data už nejsou nikdy použita. V případě aplikování několika kernelů na vstupní datový proud lze využít pomocné datové proudy mezi kernely [10]. Tuto architekturu využívá i grafická karta NVIDIA GeForce 590, která bude při praktickém testování v rámci této práce použita.

1.1.2 Oblast využití technologie CUDA

Výkon grafických karet je využíván v odvětvích technických, vědeckých, v zábavním průmyslu, v medicíně a všude tam, kde je potřeba provádět náročné výpočetní operace. Například v biomedicíně k analýze velkého objemu dat, a dále u kryptologických algoritmů a hašovacích funkcí, kde je snaha vývojářů testovat možnost rozšifrování z důvodu zvýšení bezpečnosti. Při simulacích využívajících složité funkce, například předpověď počasí.

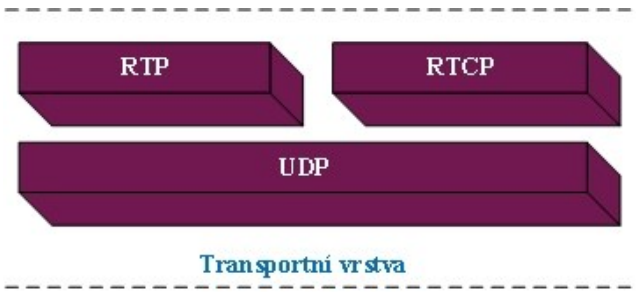
V rámci této práce bude technologie CUDA testována při zpracovávání paketů datového toku IDS/IPS Suricata v reálném čase.

1.2 VoIP

V této části se zmíním o VoIP (Voice over Internet Protocol), jelikož zatížení Suricaty bude probíhat právě s využitím testování a detekcí incidentů v IP telefonní infrastruktuře. U VoIP komunikace je vyžadována vysoká citlivost na kvalitu hovoru, který musí probíhat v reálném čase. Je však nutné zajistit, aby nedošlo například pomocí DoS (Denial of Service) k útoku na kvalitu těchto parametrů, nebo MiTM (Man in The Middle) útokem k odposlechu vzájemné komunikace. Pokud by došlo k útoku, musí být odražen v maximálně několika sekundách, to znamená dřív, než dojde k narušení služby [19]. To je jeden z důvodů, proč byla zvolena testovací topologie s IP telefonií při testování výkonnosti IDS Suricata.

1.2.1 Transportní protokoly

Transportními protokoly VoIP komunikace jsou UDP (RFC 768) [16], RTP (RFC 1889, 3550) [17, 18]. Těmito protokoly je přenášen hlas, video a jiná data spojená s administrací hovoru. Umístění transportních protokolů v modelu ISO/OSI je zobrazeno na obr. 1.4 [19].



Obrázek 1.4 *Uspořádání transportních protokolů v modelu ISO/OSI.*

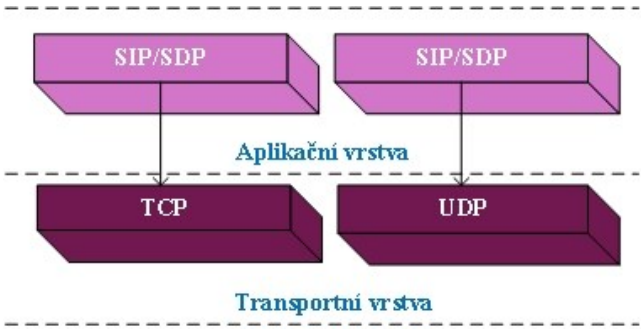
RTP (Real-time Transfer protocol) protokolem je ve VoIP přenášen hlas a obraz v reálném čase. Přenos může probíhat jako Unicast nebo Multicast, to znamená jedním odesílatelem a více příjemci. Přenos probíhá na dynamicky přidělených portech. Dokumentace tohoto bezestavového protokolu je uvedena v normě IETF RFC 1889 [17] a IETF RFC 3550 [18]. RTP přidává časovou známku a sleduje doručení datagramů, což je ale informativní zpráva, která nemá vliv na doručení datagramů. Samotný protokol neposkytuje žádné šifrování přenosu RTP paketů. Pakety je možno šifrovat rozšířením protokolu SRTP nebo ZRTP.

RTCP (Real-time Transfer Control Protocol) se používá k pravidelnému přenosu kontrolních paketů účastníkům datového proudu, kterým také poskytuje řídící informace pro datový RTP proud. Hlavní funkcí tohoto protokolu je poskytování zpětné vazby na kvalitu služeb QoS (Quality of Service) vázanou na RTP. Shromažďuje informace, jako jsou například počet odeslaných paketů a ztracených paketů, počet odeslaných Byte, jitter (kolísání zpoždění) a dobu odezvy. Tyto informace může SIP klient použít ke zvýšení kvality služeb zvýšením datového toku nebo použitím jiného kodeku [18]. Protokol RTCP není šifrovaný, k tomuto účelu může být použit protokol SRTCP.

1.2.2 Signalizační protokoly

SDP (Session Description protocol) popisuje vlastnosti relace při multimediálním přenosu. Nepřenáší vlastní data, ale je využíván k vyjednání parametrů spojení, kterými jsou typ média (audio nebo video), transportní protokol (RTP/UDP/IP, H.320, TCP), typ kodeku, přenosová rychlost. Protokol popisuje norma RFC 4566 [30]. Je používán ve spojení se SIP.

SIP (Session Initiation Protocol) je protokol IP telefonie, využívající se k zahájení, modifikaci a ukončení telefonických hovorů ve VoIP. Umístění tohoto protokolu v modelu ISO/OSI je zobrazeno na obr. 1.5 [19]. SIP je popsán v normě IETF RFC 3261. [20]. Rozšíření SIP protokolu (například Instant messaging) jsou definována v samostatných RFC.



Obrázek 1.5 *Uspořádání signalizačních protokolů v modelu ISO/OSI.*

Protokol pracuje na bázi výměny žádostí a odpovědí. Pro inicializaci relací je používán protokol UDP na portu 5060 i TCP, rovněž na portu 5060. SIP je hlavičkou paketu zasílající žádosti a odpovědi a protokol SDP je tělem paketu, ve kterém jsou obsaženy informace k zahajovanému přenosu. Samotná data jsou při hovoru přenášena protokolem RTP.

Metody SIP protokolu definované v RFC 3261 [20]:

INVITE – slouží k navazování spojení nebo u již existujícího spojení ke změně parametrů,
ACK – potvrzení o obdržení požadavku,
CANCEL – žádost k přerušení zahajované relace ještě před jejím navázáním,
REGISTER – žádost k registraci účastníka na registračním serveru,
OPTIONS – požádá o informace o možnostech volajícího, aniž by se sestavilo volání,
BYE – žádost o ukončení relace.

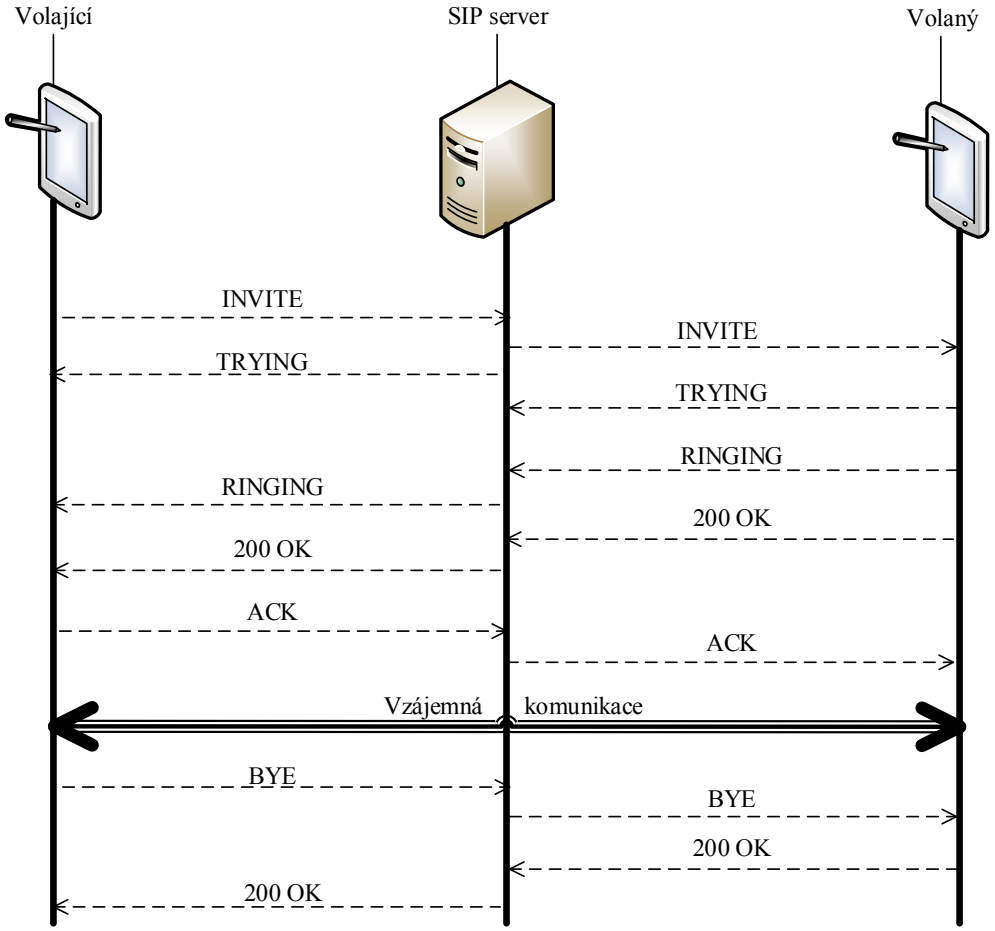
Metody definované v dalších RFC [20]:

REFFER – požadavek jiného uživatele k řízení spojeni – RFC 3515 [21],
SUBSCRIBE – požadavek pro zjištění stavu vzdáleného prvku – RFC 3265 [22],
NOTIFY – k informování koncového prvku – RFC 3265 [22],
UPDATE – aktualizace o stavu relace – RFC 3331 [23],
INFO – k přenosu informace během relace – RFC 2976 [24],
PRACK – potvrzení dočasné odpovědi – RFC 3262 [25],
MESSAGE – pro instant messaging – RFC 3428 [26].

Chybová hlášení a odpovědi protokolu mají vedle číselného označení také textovou verzi, což je uvedeno v doporučení RFC 3261 [20][19]:

1xx – informační odpovědi, kdy byl požadavek obdržen a zpracovává se
(100 - Trying, 180 - Ringing, 182 - Queued),
2xx – kladné odpovědi o tom, že požadavek byl úspěšně obdržen a zpracován bez problémů
(200 – OK, 202 - Accepted),
3xx – odpovědi o přesměrování
(300 – Multiple Choices, 301 - Moved Permanently, 302 – Moved Temporarily, 305 Use Proxy),
4xx – chyba na straně klienta, požadavek je chybný a nemůže být serverem zpracován
(400 – Bad Request, 403 – Forbidden, 483 – Too Many Hops),
5xx – požadavek je v pořádku, ale chyba je na straně serveru
(500 – Server Internal Error, 504 – Server Time-out, 513 – Message Too Large),
6xx – globální chyba
(600 – Busy Everywhere, 603 – Decline, 604 – Does Not Exist Anywhere).

Relace mezi SIP klienty se navazuje přímo mezi sebou, nebo pomocí i několika SIP proxy serverů, které mohou plnit funkci SIP registrátora, kde se účastníci registrují. SIP pakety nejsou zabezpečeny. Tím se účastníci SIP relace vystavují nebezpečí zjištění třetí osobou názvu účtu účastníků, včetně dané IP adresy, jakého používáme SIP klienta, a jiné podrobnosti. Proto je nutno komunikaci zabezpečit. Navázání komunikace mezi dvěma klienty je názorně ukázáno na obrázku 1.6.



Obrázek 1.6 *Navázání komunikace na protokolu SIP.*

Architektura SIP

***SIP adresa (SIP URI)*:** Identifikuje uživatele. Má podobný tvar, jako e-mailová adresa s tím rozdílem, že obsahuje předponu SIP (*sip:*uživatel@host). V adrese mohou být zahrnuty i nepovinné údaje, jako je port nebo parametry URI. Pokud nepovinné údaje nejsou uvedeny, předpokládá se u nich použití všeobecně známých hodnot. Například u portu se předpokládá použití portu 5060. Při vytvoření šifrovaného připojení je místo předpony *sip* uvedena předpona *sips* [19] [27].

***Transakce*:** Je metoda požadavku a všech odpovědí, které na tento požadavek odpovídají. Příkladem je BYE a 200 OK [27].

Dialog: Je posloupnost SIP zpráv mezi dvěma UA, které mají vzájemnou souvislost [27].

***SIP Server*:** Má většinou zodpovědnost za uživatele v doméně. Pracuje v módu *redirect* nebo *proxy*. V módu *redirect* sdělí SIP server adresu volajícího volanému, který dále navazuje spojení přímo na novou adresu. V *proxy* módu se spojení navazuje prostřednictvím serveru [27].

***Registrar*:** Je registrační server, který zpracovává požadavek REGISTER a získané informace o poloze uživatele předá lokalizační službě [27].

***Location Service*:** Tato služba poskytuje informace o aktuální IP adrese uživatele. K tomu může využít buď Radius protokol, nebo databázi s kontakty [27].

***Transaction stateful proxy*:** Udržuje stav transakce po celou dobu od přijetí požadavku, do odeslání konečné odpovědi [27].

***Call statefull proxy*:** Udržuje stav dialogu od prvního požadavku INVITE, až do ukončovacího požadavku BYE [27].

Hlavičky SIP

To: Adresa volaného,
From: Adresa volajícího,
Via: Adresa volajícího, a pokud požadavek na spojení prošel serverem, tak i jeho adresa. Stejnou cestou se bude vracet i odpověď,
Call-Id: Unikátní identifikace volání,
Contact: Aktuální skutečná adresa,
Record-Route: Obsahuje adresy serverů, které veškerou komunikaci náležící k hovoru chtějí dostávat,
Route: Postupně uvedené adresy od volajícího k volanému, včetně adres serverů, přes které je požadavek směrován,
Request-URI: Aktuální adresa volaného, která je uvedena jako první za typem metody.

2 Intrusion Detection System

IDS systémy byly vyvinuty za účelem ochrany počítačové sítě a v případě nestandartního chování v síti o tomto například informuje pověřenou osobu. IDS systém může být realizován jako hardware nebo software. V této kapitole bude konkrétně přestaven open-source IDS od společnosti OISF (Open Information Security Foundation) Suricata, který bude použit v praktické části diplomové práce.

Suricata je velmi výkonný síťový IDS a IPS. Je Open Source a je spravována neziskovou organizací OISF. Od 27. ledna 2016 je ke stažení stabilní verze 3.0. Suricata je vysoce škálovatelná, dokáže paralelizovat zátěž do více vláken. Výkon je v reálném čase měněn tak, aby nebyl zbytečně nákladný při nižším zatížení a zase naopak, aby nebyl výkon příliš slabý při velkém zatížení sítě. Suricata dokáže na základě předdefinovaných pravidel zvládat v reálném čase rychlost až 10 Gb/s. Běžné protokoly Suricata automaticky rozpozná, což zjednodušuje psaní pravidel. Také umí nalézt klíčová slova na mnoha protokolech od HTTP URI, až po identifikátor SSL certifikátu. Stejně tak rozeznává různé typy souborů. [11].

Základním konfiguračním souborem IDS/IPS Suricata je `suricata.yaml`. V tomto souboru lze nastavit chování Suricaty (IDS nebo IPS), jak velkou část systémových prostředků může Suricata využít v různých nastaveních, využití technologie CUDA a další. Možnosti nastavení tohoto konfiguračního souboru jsou uvedeny v příloze A.

2.1 Více vláknová podpora

Suricata využívá za účelem detekce více vláken, to znamená, že se snaží využít maximální počet jader procesorů. Tím se nabízí možnost využití tohoto potenciálu u grafických karet podporujících tuto komunikaci společnosti NVIDIA s označením CUDA jako hardware, nebo Open-GL. Open-GL je multiplatformní, čímž je myšleno, že lze použít tuto více vláknovou technologii u grafických karet, které ji hardwarově nepodporují pomocí softwarové nadstavby v systémech, které umožňují práci s grafikou. Více vláknová podpora s využitím grafických karet je ideálním řešením, jelikož jádra procesorů grafických karet umožňují rychle provést jednoduché operace.

2.2 Softwarové IDS

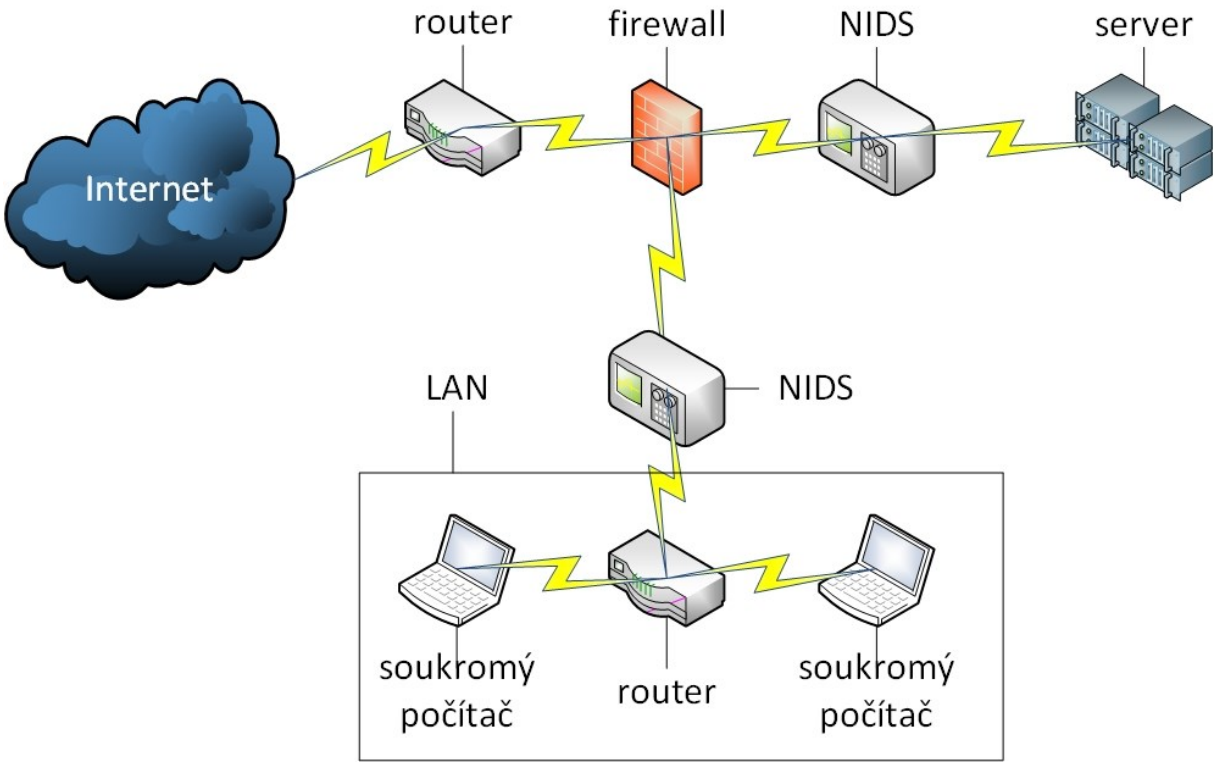
IDS byly vyvinuty k ochraně počítačové sítě před hrozbami z veřejné dostupné sítě, a také k ochraně před hrozbami uvnitř privátní sítě. Slouží jako pasivní prvek, to znamená, že informuje o podezřelém chování například zápisem do log souboru. Jako aktivní prvek slouží IPS, který může blokovat provoz nebo filtrovat pakety. Úlohu IDS systému je třeba brát velice vážně, jelikož významně napomáhá k odhalení ať už náhodného, nebo cíleného útoku (například získání e-mailů nebo rodných čísel). Správně nakonfigurovaný IDS (popř. IPS) spolu s použitím šifrováním dat (skrytí obsahu zprávy), blokováním přístupu do sítě pomocí firewallu, použitím Honeypotu (sledování aktivit útočníka za účelem jeho odhalení, nebo vytvoření obrany dle znalosti způsobu útoku), antiviru, popř. zamezení přístupu do

samotné sítě pomocí VLAN nebo VPN, významně zvýší zabezpečení počítačové sítě, čímž lze na minimum snížit počet útoků a věřím, že od některých bude raději upuštěno.

IDS lze zapojit do různých částí sítě v závislosti, kterou její strategickou část chceme ochránit, případně i celou síť jedním IDS, pokud síť není příliš rozsáhlá. Podle způsobu zapojení jsou definovány základní typy IDS, kterými jsou NIDS (Network Intrusion Detection System), HIDS (Host-Based Intrusion Detection System) a DIDS (Distributed intrusion Detection System).

2.2.1 NIDS

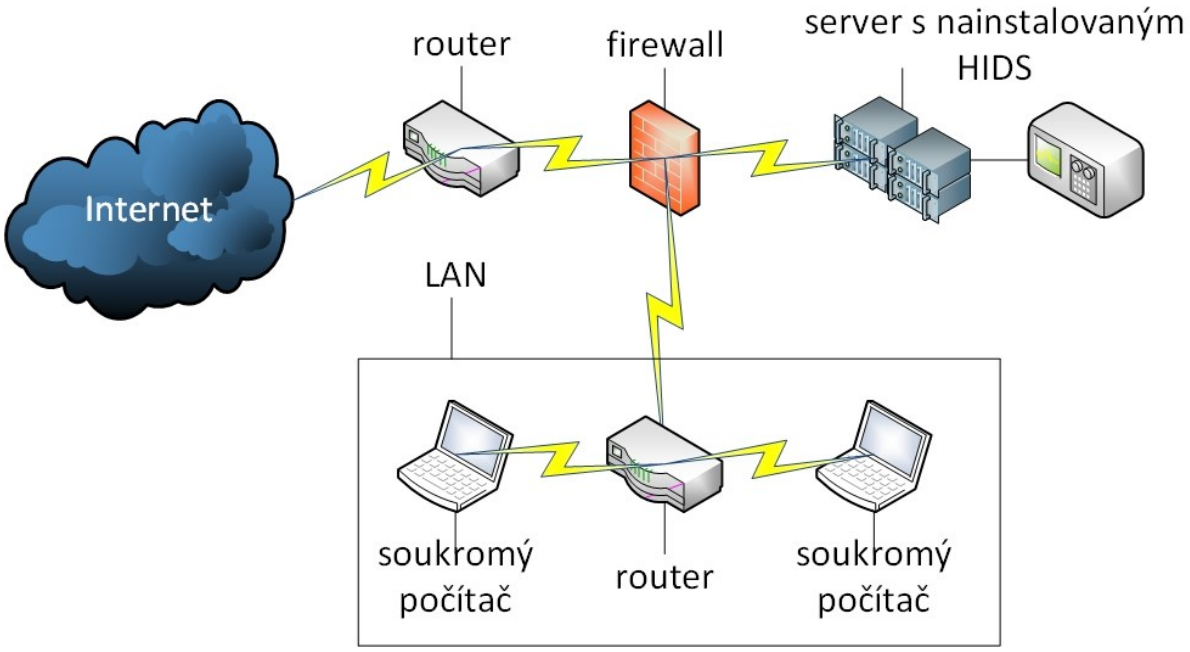
Tento typ zapojení IDS je implementován v síti tak, aby ji chránil celou, to znamená, že je zapojen na některém z aktivních síťových prvků jako je například směrovač nebo přepínač, viz obrázek 2.1. Výhodou tohoto zapojení je, že dokáže monitorovat veškerý provoz na sub-síti, takže odhalí útok na síťovou vrstvu, kde sleduje parametry TCP/IP hlaviček. S tímto však souvisí i vysoká zatíženost NIDS, kdy může dojít k jeho přetížení a také nemá možnost kontrolovat aplikační vrstvu ISO/OSI s čímž souvisí to, že nedokáže rozpoznat povahu útoku nebo na jakou aplikaci je útok soustředěn [12].



Obrázek 2.1 *Ukázka síťové topologie NIDS.*

2.2.2 HIDS

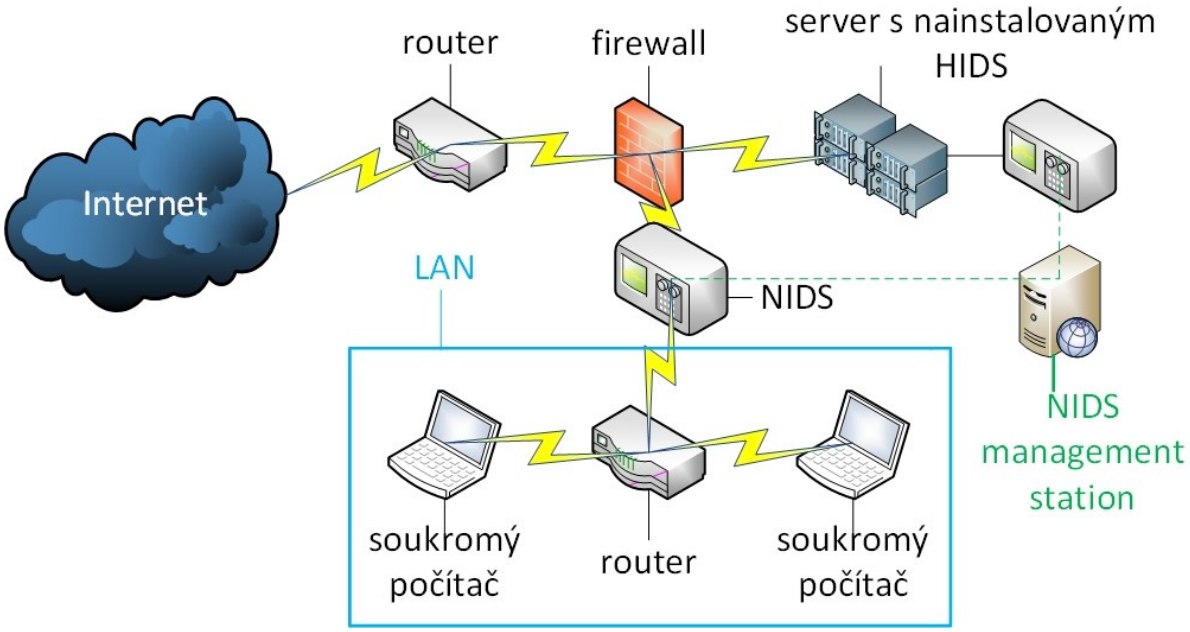
Tento způsob implementace IDS je součástí hostitelského systému, to znamená je na něm oproti NIDS závislý. Toto je z jedné stránky omezující pro jeho variabilitu nasazení, ale dokáže v porovnání s NIDS reagovat daleko lépe na útoky, jelikož ví, jak hostitelský systém pracuje a i jednotlivé aplikace systému, viz obrázek 2.2. Další jeho výhodou je, že sleduje i zašifrovanou komunikaci. Pracuje tedy na aplikační vrstvě ISO/OSI. S tím však také souvisí nemožnost rozpoznat útok na nižší vrstvě ISO/OSI a také není imunní proti DoS (nebo DDoS) útoku a při přetížení systému přestává pracovat stejně jako napadený systém [12].



Obrázek 2.2 *Ukázka síťové topologie HIDS.*

2.2.3 DIDS

Jedná se o spojení výhod obou předešlých systémů NIDS a HIDS. Na počítačové síti, kterou chceme chránit je rozmístěno více IDS v podobě HIDS a NIDS v závislosti, jakou část nebo síťovou vrstvu ISO/OSI chceme mít pod kontrolou. Tyto IDS by se daly nazvat jakýmisi senzory, které jsou ovládány centrálním NIDS, viz obrázek 2.3. Na jednotlivých senzorech lze nastavit různá pravidla a v případě detekce průniku jsou výsledky odesílány na centrální NIDS. Senzory mohou z centrálního NIDS nahrávat aktualizace. Tímto způsobem lze tedy rovnoměrněji sledovat celou síť a jsou tedy i k dispozici přesnější výsledky nalezených hrozeb, čímž lze efektivněji na nežádoucí průniky do sítě reagovat [12].



Obrázek 2.3 *Ukázka síťové topologie DIDS.*

Suricata využívá za účelem detekce více vláken, to znamená, že se snaží využít maximálně počet jader procesorů. Tím se nabízí možnost vyžití tohoto potenciálu u grafických karet podporujících tuto komunikaci společnosti NVIDIA s označením CUDA jako hardware nebo open GL. Open GL je multiplatformní, čímž je myšleno, že lze použít tuto více vláknovou technologii i u grafických karet, které ji hardwarově nepodporují pomocí softwarové nadstavby v systémech, které umožňují práci s grafikou. Více vláknová podpora s využitím grafických karet je ideálním řešením, jelikož jádra procesorů grafických karet umožňují rychle provést jednoduché operace.

3 Přehled nástrojů pro penetraci v IP telefonii

Přenos digitalizovaného hlasu počítačovou sítí vyžaduje vysokou citlivost na kvalitu hovoru, který musí probíhat v reálném čase. Aby byl přenos hlasu bezpečný, musí být komunikace zajištěna proti různým typům útoků. Tím je myšlena ochrana proti záměru útočníka, například DoS útokem snížit kvalitu hovoru nebo i jeho přerušení, nebo zabránit MiTM útoku směřujícího k odposlechu vzájemné komunikace. Případný útok musí být zastaven v max. několika sekundách, než dojde k narušení služby.

Proto je kladen velký důraz na rychlost analýzy síťového provozu, která má případnému útoku na VoIP telefonii zabránit. K ochraně sítě, ve které probíhá přenos hlasu RTP, nebo SRTP pakety značnou mírou přispívá i její analýza IDS, nebo IPS, kdy kvalita zabezpečení, jak bylo uvedeno výše, úzce souvisí s rychlostí vyhodnocení rizik, kdy je kladen důraz, aby nebyl touto analýzou datový tok omezen.

Suricata může analyzovat datový provoz i s využitím CUDA technologie, která má zvýšit výkon intrusivního detekčního systému. V této části diplomové práce budou představeny různé penetrační nástroje. Tyto se rozdělují na dvě základní skupiny a to jsou pasivní penetrační nástroje, které slouží k odposlechu komunikace v síti, aniž by bezpečnostní mechanismy měly možnost tento odposlech zjistit. Tyto nástroje slouží k získání údajů z provozu sítě, které jsou poté aktivními penetračními nástroji využity k útoku, který má za cíl snížit kvalitu datového přenosu, přerušit datový tok nebo způsobit například nedostupnost PBX ústředny. Další z aktivních penetračních nástrojů má za cíl, po získání uživatelského jména a IP adresy telefonní ústředny, pomocí slovníkového útoku zjistit přihlašovací údaje k účtu účastníka a tím tento účet poté ovládnout.

Pro přehled je tato kapitola rozdělena na penetrační nástroje, které jsou vhodné k testování výkonnosti IDS/IPS Suricata a které ne. Nevhodné nástroje jsou zde však také uvedeny a to z důvodu získání uceleného přehledu.

3.1 Pasivní penetrační nástroje

Tyto penetrační nástroje jsou vhodné k získávání důležitých informací z provozu sítě a tyto informace jsou poté využity aktivními penetračními nástroji, jejichž cílem je znemožnit správnou funkci sítě způsobem, kdy dojde k odmítnutí poskytované služby nebo k její významnému zhoršení kvality.

Ve VoIP jsou tyto nástroje zejména využity k získání IP adresy sítě, IP adresy telefonní ústředny, portu na kterém komunikuje a k jejímu názvu. Dále slouží k získání IP adresy účtů telefonní ústředny, jejich názvů a i k zjištění typu zařízení, ze kterých dochází k přihlašování na telefonní ústřednu. Dále lze zjistit protokol, na kterém telefonní ústředna komunikuje.

Níže budou představeny penetrační nástroje, které slouží k zjišťování potřebných informací ze sítě za účelem provedení aktivního útoku a také nástroje, které bylo vhodné představit, ale v praktické části nebyly použity z důvodu, že tato diplomová práce je zaměřena především na výkonové testování zatížení IDS/IPS Suricata ve VoIP.

3.1.1 Ettercap

Jedná se o nástroj umožňující MiTM útok. V době vydání diplomové práce (únor 2016) je k dispozici verze 0.8.2, která byla taktéž použita při testování výkonnosti Suricaty.

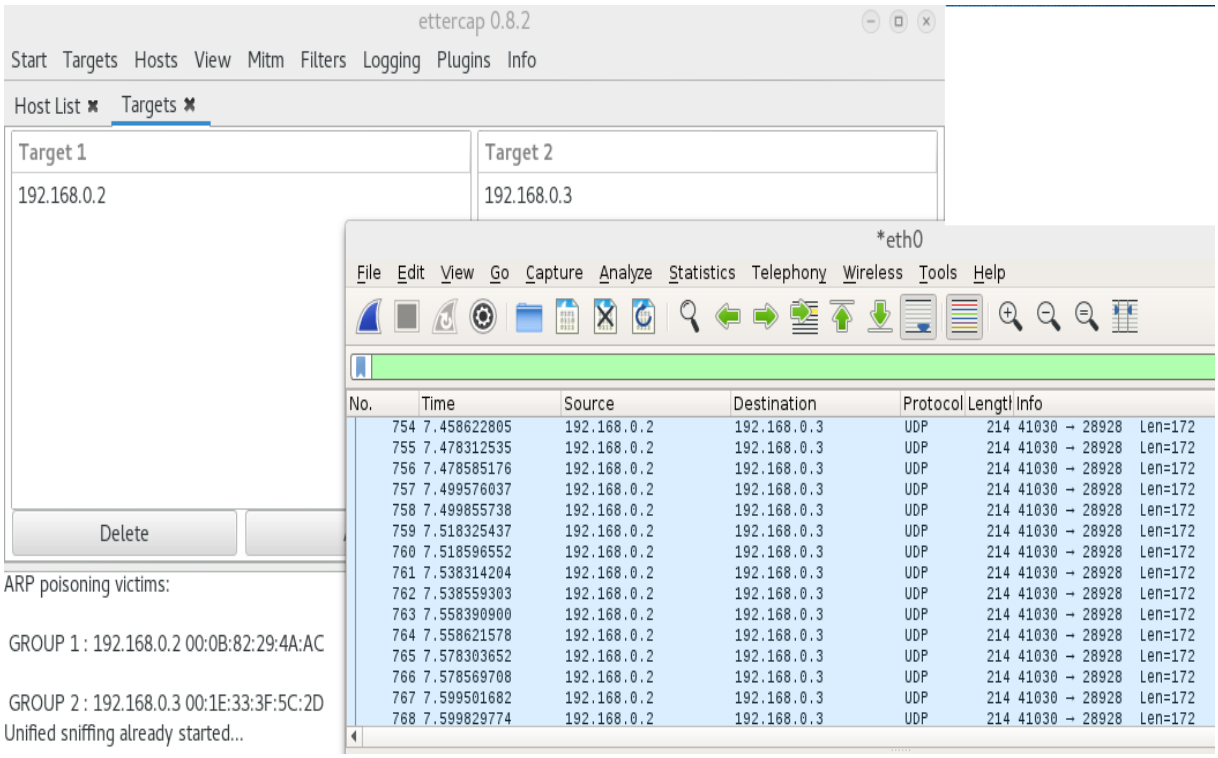
Program obsahuje několik Man-in-the-middle útoků. Dokáže zachytit data aktivních spojení, odchytit HTTPS hesla a filtrovat provoz v síti.

Ettercap je spustitelný těmito způsoby:

- *graficky* s použitím parametru `ettercap -G`
- *textově* s použitím parametru `ettercap -T`
- *v příkazové řádce* s použitím parametru `ettercap -C`.

Nástroj Ettercap dokáže najít všechny aktivní IP adresy, což si můžeme zkontrolovat příkazem `ifconfig`.

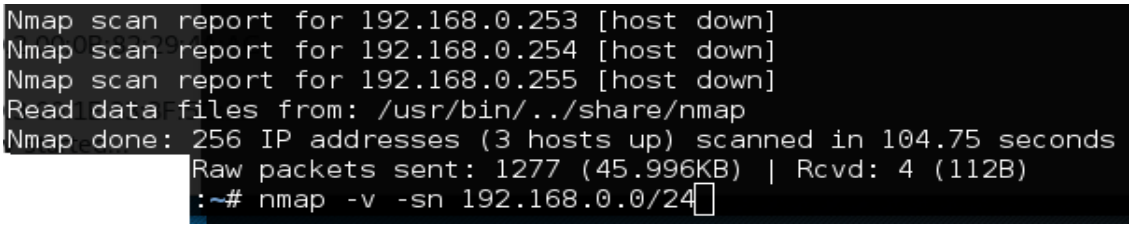
Ettercap dále umí ARP poisonning, DHCP Spoofing, port stealing. Žádný z těchto nástrojů však delší dobu kontinuálně nezatěžuje oběť útoku a pro náš test výkonnosti Suricaty s CUDA a bez CUDA není vhodný a proto tento nástroj nebyl v praktické části použit. Praktická ukázka z testování je vidět na obrázku 3.1.



Obrázek 3.1 *Ukázka grafického rozhraní penetračního nástroje Ettercap.*

3.1.2 nmap

V době vydání diplomové práce (únor 2016) je tento nástroj ve verzi 7. Jedná se nástroj sloužící k získávání informací o síti, to znamená o aktivních spojeních, operačních systémech počítačů, spuštěných službách, jaké druhy paketů daný počítač odesílá do sítě a přijímá [37], viz obrázek 3.2.



Obrázek 3.2 *Ukázka ze skenování nástrojem nmap.*

3.1.3 VoIPhopper

VoIPhopper je nástroj vytvořený v programovacím jazyku C, který rychle pracuje ve Voice VLAN sítích na specifikovaných portech. VoIPhopper napodobuje chování IP telefonů výrobců Cisco, Avaya, Nortel a prostředí Alcatel-Lucent. Nástroj slouží k testování zabezpečení VoIP infrastruktury [40].

Nástroj pracuje ve dvou důležitých krocích. Prvním je nalezení 12 bitového Voice VLAN ID (VVID), které používají IP telefony. Tento identifikátor VoIPhopper dokáže nalézt v protokolech CDP, DHCP, LLDP-MED, 802.1q ARP [40].

Dále nástroj vytvoří virtuální VoIP Ethernetové rozhraní na operačním systému. Poté vloží falešnou 4 Byte 802.1q hlavičku obsahující 12 bitové VLAN ID do falešného DHCP požadavku. Poté co obdrží IP adresu ve VoIP VLAN podsíti, budou následně všechny Ethernetové rámce označeny falešnou 802.1q hlavičkou [40].

VoIPhopper je určen pro telefony a infrastrukturu firem Cisco, Avaya, Alcatel-Lucent a Nortel. Jelikož žádná z výše uvedených firem není v testovací topologii zapojena, nebude tento nástroj použit v praktické části.

3.1.4 OpenVAS

OpenVAS (Open Vulnerability Assessment System) je sada nástrojů určená k penetračnímu testování. V době vydání diplomové páce je ve verzi 8 (únor 2016). Využívá pravidelně aktualizovanou databázi NVT (Network Vulnerability Tests) [41].

OpenVAS obsahuje manager (řídí a nastavuje testy, spravuje uživatele, synchronizuje databázi), scanner (testuje zařízení), Greenbone Security Assistant (ovládání OpenVAS webovým klientem) a OpenVAS CLI (ovládání OpenVAS z příkazové řádky) [41].

OpenVAS je předinstalován v distribuci Kali-linux. Tento soubor nástrojů odhaluje chyby, jako je například neaktuální verze operačního systému, výchozí přihlašování pod účtem správce,

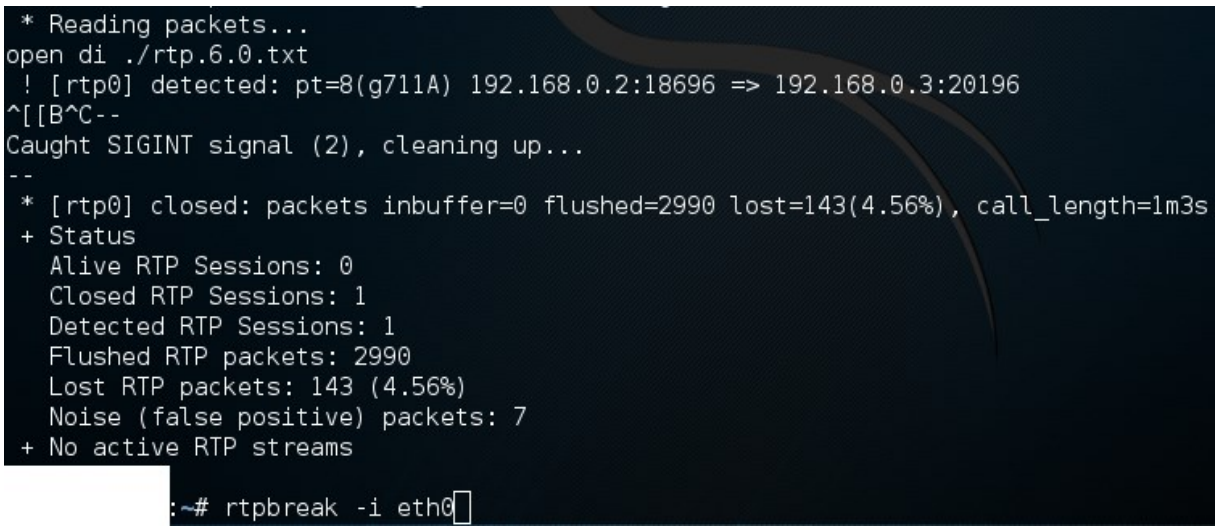
nezabezpečené přihlašování FTP, zranitelnosti serveru Apache. Tento nástroj však není vhodný pro testování zatížitelnosti IDS s technologií CUDA a bez ní, proto nebyl v praktické části použit.

3.1.5 OhrWurm

Je to jednoduchý nástroj, který dokáže číst SIP zprávy mezi dvěma SIP účty. Tímto nástrojem lze získat informací o číslech RTP portů. Dále dokáže potlačit RTCP pakety. Podmínkou funkčnosti tohoto nástroje je, aby komunikace mezi dvěma SIP klienty probíhala v přepínané LAN síti. OhrWurm sám o sobě nedokáže provádět MiTM útok. K tomuto potřebuje nástroj arpspoof ze sady nástrojů dsniff (kolekce nástrojů pro testování bezpečnosti sítě) [41]. Arpspoof umožňuje odeslat konkrétnímu počítači bezdůvodnou ARP odpověď s podvrženou IP adresou. Tímto lze veškerý provoz směrovat skrz počítač útočníka a poté nástroj OhrWurm již může číst SIP zprávy mezi dvěma SIP účty. V době vydání diplomové práce (únor 2016) je ve verzi 0.1.

3.1.6 rtpbreak

Tento nástroj slouží k nalezení RTP provozu v síti, který následně umí rekonstruovat a analyzovat. Nevyžaduje přítomnost RTCP paketů a pracuje nezávisle na signalizačních protokolech [45]. Pracuje jak na paketové, tak bezdrátové síti. Tento nástroj pasivně zachytává RTP pakety a proto není vhodný k zátěžovému testování a nebude v praktické části použit. Praktická ukázka je na obrázku 3.3. V době vydání diplomové práce (únor 2016) je ve verzi 1.3a.



Obrázek 3.3 *Praktická ukázka z testování penetračního nástroje rtpbreak.*

3.1.7 Xplico

Tato aplikace extrahuje z internetového provozu pakety těchto protokolů: POP, IMAP, SMTP, SIP, MGCP, H323, FTP, TFTP a další. Získané informace zobrazuje v SQLite nebo Mysql databázi. Ukládané jsou ve formátu XML. Úplnost získaných paketů dokáže opět zkontrolovat pomocí ACK odpovědí [50] [51].

Tento nástroj neprovádí aktivně žádné útoky, ale sestavuje datový tok paketů různých protokolů. V době vydání diplomové práce (únor 2016) je ve verzi 1.1.2.

3.1.8 john the ripper

Nástroj v sobě kombinuje několik aktivních penetračních nástrojů, které jsou plně konfigurovatelné k potřebám účelu, ke kterému mají být použity. Hlavně se jedná o prolamování zahašovaných hesel. John the Ripper podporuje tyto hash: Windows NTLM (MD4-based), Mac OS X 10.4-10.6 salted SHA-1 hashes, Mac OS X 10.7 salted SHA-512 hashes, raw MD5 and SHA-1, libovolné MD5-based webových aplikací, hash používané SQL databázovými servery (MySQL, MS SQL, Oracle) a některými LDAP servery. Dále některé typy hash používané OpenVMS, privátní klíče OpenSSH, Kerberos TGTs, PDF soubory, ZIP (klasické PKZIP a WinZip/AES) a RAR archivy [53]. V době vydání diplomové práce (únor 2016) je ve verzi 1.8.0.6.

3.1.9 Ace-VoIP

Ace (Automated Corporate Enumerator) – VoIP napodobuje chování IP telefonu. Tento penetrační nástroj se snaží být plně automatizovaný. Jako první je třeba určit rozhraní sítě VLAN. Za další je na výběr MiTM mód nebo Monitor mód. V MiTM módu je na výběr, zda se bude penetrační nástroj snažit zjistit informace o uživatelích sítě nebo pokud známe konkrétního uživatele, tak podniknout penetrační test na konkrétním uživateli [52].

Nástroj tedy aktivně nezasahuje do síťového provozu, ale zjišťuje z jejího provozu informace o její topologii a uživatelích. Tento nástroj tedy není vhodný k zátěžovému testování IDS. V době vydání diplomové práce (únor 2016) je ve verzi 1.10-1.

3.1.10 EnumIAX

Slouží k prolamování hesel k účtům telefonní ústředny Asterisk pomocí slovníkových útoků na protokolu IAX [54]. Testovací topologie byla vytvořena s telefonní ústřednou Asterisk, ale na protokolu SIP, proto tento nástroj nebyl v praktické části použit. V době vydání diplomové práce (únor 2016) je ve verzi 0.4a.

3.2 Aktivní penetrační nástroje

Níže uvedené nástroje jsou vhodné k zátěžovému výkonnostnímu testování IDS/IPS Suricata z důvodu, že kontinuálně zatěžují nějakým způsobem síť, a to buď neustálými různými žádostmi za účelem vyčerpání systémových prostředků serveru s nainstalovanou telefonní ústřednou, nebo slovníkovými útoky za účelem zjištění přístupových údajů k telefonnímu účtu.

Jak bylo uvedeno v úvodu, je velice důležité zabránit těmto útokům, co možná v nejkratším čase, aby nedošlo k narušení přenosu hlasu, který probíhá v reálném čase a je citlivý na zpoždění, ke kterému by po vyčerpání systémových prostředků došlo, nebo by mohlo dojít i k odmítnuté služby.

V rámci diplomové práce budou k zátěžovému testování IDS/IPS Suricata použity níže uvedené penetrační nástroje. Jejich konfigurace a příkazy, kterými byly spuštěny, budou uvedeny v praktické části.

3.2.1 InviteFlood

Tento nástroj je typu DoS útoku, kdy zasílá neustále falešnou žádost o spojení se SIP serverem, což vede k jeho neschopnosti zpracovávat jiné požadavky. Tento nástroj je vhodný k testování zatížení Suricaty s CUDA a bez CUDA. V době vydání diplomové práce (únor 2016) je k dispozici verze 2 [36].

Zde na obrázku 3.4 je uvedena ukázka syntaxe penetračního nástroje InviteFlood použitá v praktické části kapitoly 5.

```
inviteflood eth0 2000 192.168.0.1 192.168.0.3 100000000
```

Obrázek 3.4 *Ukázka syntaxe penetračního nástroje InviteFlood.*

3.2.2 SIPp

V době vydání diplomové práce (únor 2016) je aktuální verze 3.4. SIPp umí vytvářet žádosti REGISTER, OPTIONS, INVITE a lze testovat odpovědi: ACK nebo BYE [38]. SIPp je hlavně využíván pro generování hovorů, jejichž jednotlivé parametry lze předem nadefinovat. Ve scénářích je možno simulovat vzájemnou komunikaci SIP klientů nebo simulovat několik telefonních hovorů současně. Z důvodu variability možnosti nastavení různých scénářů byl tento nástroj použit v praktické části.

3.2.3 SIPvicious

Jedná se o sadu 5 nástrojů *svmap, svwar, svcrack, svreport, svcrash* [39]. V době vydání diplomové práce (únor 2016) je ve verzi 0.2.8.

- ***svcrack*** - slouží k prolamování hesel na telefonní ústředně pomocí slovníkového útoku,
- ***svcrash*** - zastavuje neautorizované spuštění *svwar* a *svcrack*,
- ***svreport*** - spravuje události a exportuje hlášení do formátů pdf, xml, csv nebo txt,

- ***svmap*** - vyhledává SIP zařízení v definovaném IP rozsahu,
- ***svwar*** - vyhledává aktivní pravidla na telefonní ústředně.

Ze sady výše uvedených nástrojů je v praktické části použit nástroj *svcrack* pro jeho vhodnost k zátěžovému testování IDS Suricata.

3.2.4 rtpflood

Rtpflood je aktivní nástroj zaplavující RTP pakety jakékoliv zařízení. Tento nástroj je ovládán z příkazové řádky [46]. K úspěšnému provedení testu musí být znám port, na kterém probíhá přenos RTP paketů. V době vydání diplomové práce (únor 2016) je ve verzi 1.0-1. Praktická ukázka syntaxe je uvedena na obrázku 3.5. Zde je jako první uvedena IP adresa zdroje provozu, poté IP adresa cíle provozu RTP paketů, dále port zdroje a port cíle. Číslo 1000 uvádí, že bude zasláno 1000 paketů se sekvenčním číslem 3, které využívají RTP pakety pro identifikaci jednotlivých RTP toků v síti. V normální síti jsou čísla dynamicky generována. Zde je možnost je zadat ručně.

```
rtpflood 192.168.0.4 192.168.0.3 5060 5061 1000 3
```

Obrázek 3.5 *Praktická ukázka syntaxe nástroje rtpflood.*

3.2.5 rtpinsertsound

Jedná se o nástroj, který vkládá audio soubor do RTP streamu [47]. V době vydání diplomové práce (únor 2016) je ve verzi 2.0.

3.2.6 rtpmixsound

Jedná se o nástroj podobný předchozímu rtpinsertaudio. Tento nástroj míchá audio soubory v reálném čase s cíleným RTP datovým proudem [48]. Praktická ukázka syntaxe příkazu je na obrázku 3.4. V době vydání diplomové práce (únor 2016) je ve verzi 3.0.

```
rtpmixsound stapler.wav -a 192.168.0.4 -A 5060 -b 192.160.0.3 -B 5061
```

Obrázek 3.6 *Praktická ukázka syntaxe penetračního nástroje mixsound.*

3.2.7 THC-Hydra

Tento nástroj má za cíl demonstrovat penetračním testem, jak snadné může být získat neoprávněný přístup k systému na dálku [43]. Podporuje Cisco AAA, Cisco auth, Cisco enable, CVS, FTP, HTTP(S)-FORM-GET, HTTP(S)-FORM-POST, HTTP(S)-GET, HTTP(S)-HEAD, HTTP-Proxy, ICQ, IMAP, IRC, LDAP, MS-SQL, MySQL, NNTP, Oracle Listener, Oracle SID, PC-Anywhere, PC-NFS, POP3,

PostgreSQL, RDP, Rexec, Rlogin, Rsh, SIP, SMB(NT), SMTP, SMTP Enum, SNMP v1+v2+v3, SOCKS5, SSH (v1 and v2), SSHKEY, Subversion, Teamspeak (TS2), Telnet, VMware-Auth, VNC and XMPP [43] [44].

Tento nástroj testuje, jak silná je autentizace výše uvedených aplikací nebo protokolů, včetně protokolu SIP pomocí slovníkových útoků. V době vydání diplomové práce (únor 2016) je ve verzi 8.1-1+b1.

4 Vytvoření testovací topologie, instalace a konfigurace

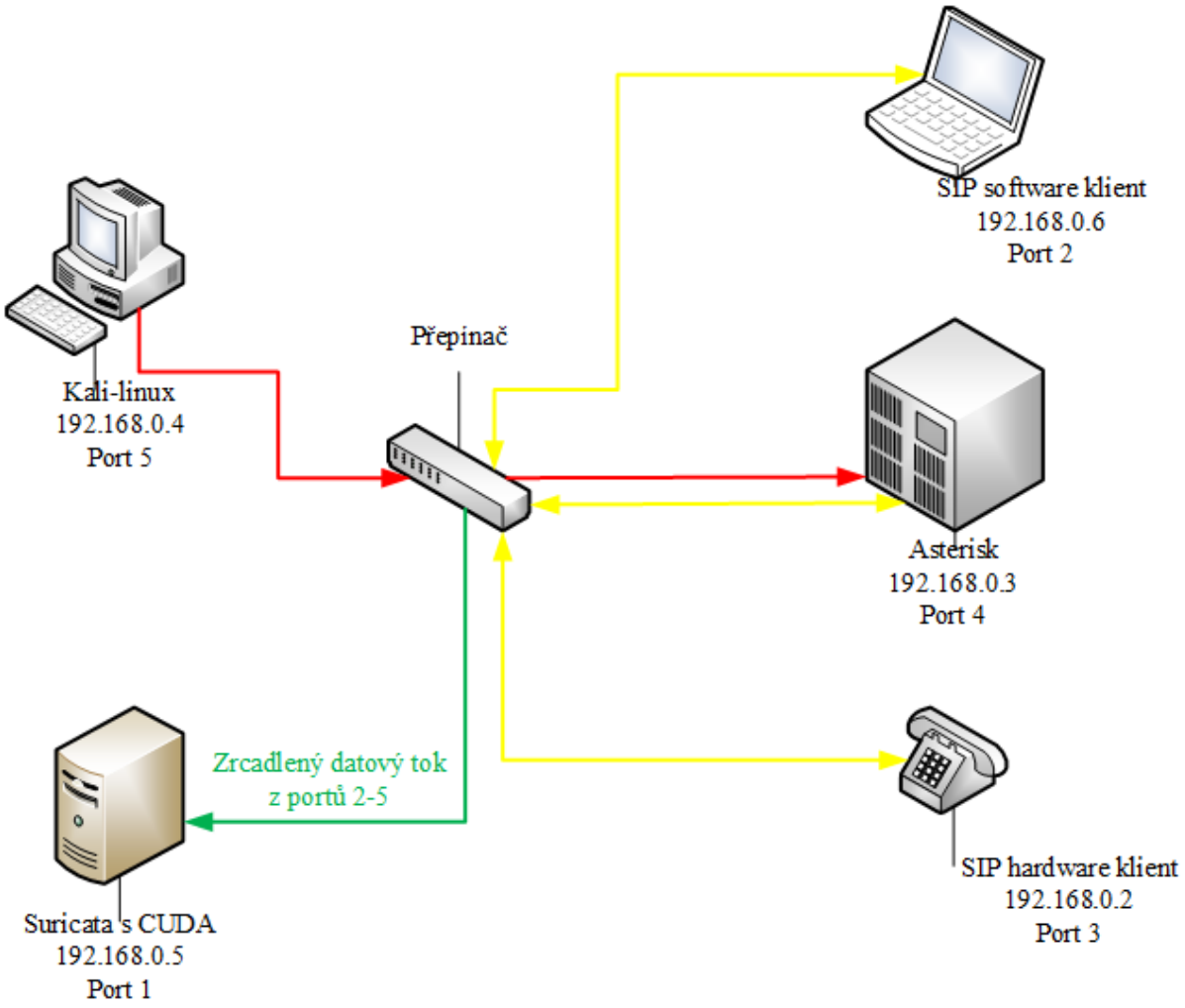
Na základě rešeršního výzkumu uvedeného v předchozí kapitole, kde byly zvoleny vybrané penetrační nástroje pro generování bezpečnostních hrozeb, bude v této kapitole objasněna testovací topologie obsahující IP telefonní prvky zabezpečené IDS/IPS Suricata. Dále zde bude uveden postup instalace potřebných aplikací a ovladačů pro správný běh a nastavení konfiguračních souborů.

4.1 Testovací topologie

Testovací topologie byla navržena v síti s podporou IP telefonie. Pobočková ústředna Asterisk byla nainstalovaná na fyzickém PC s operačním systémem Ubuntu 14.04. Tato telefonní ústředna je zapojena do přepínače TP-LINK TL-SG105E u kterého lze nakonfigurovat zrcadlení provozu z určitých portů přepínače, viz obrázek 4.1 [14]. Na tomto obrázku lze vidět, že provoz na portech 3, 4 a 5 (podbarveny modře) je zrcadlen na port 1.

Obrázek 4.1 *Mirroring a mirrored port TP-LINK.*

Do přepínače je dále zapojen IP telefon zn. GRANDSTREAM typ GXV3140 [15] a PC s operačním systémem Ubuntu 14.04 na kterém je nainstalován SIP software klient YateClient verze 4.3.1. Tyto porty (Asterisk a 2xSIP klient) jsou zrcadleny na port na kterém je zapojen IDS Suricata ver. 2.0.11. Do přepínače je dále zapojeno PC s operačním systémem Kali-linux ver. 2016-1, který představuje útočníka s komplexní sadou nástrojů k testování zabezpečení IP telefonie. Testovací topologie je názorně zobrazena na obr. 4.2.



Obrázek 4.2 *Testovací topologie.*

4.2 Konfigurace telefonní ústředny a koncových IP telefonů

Za účelem testování byla na fyzickém PC s operačním systémem Ubuntu 14.04 nainstalována open source telefonní ústředna Asterisk verze 13.7.2, která podporuje protokol SIP. Instalace byla provedena pomocí kompilace zdrojových kódů, provedením níže uvedených příkazů.

Instalace kernel hlaviček:

```
apt-get install linux-headers- "uname -r"
```

další balíčky potřebné pro kompilaci Asterisku:

```
apt-get install libssl-dev ncurses-dev
```

stáhnutí zdrojových kódů z repozitářů Asterisku:

```
wget http://downloads.asterisk.org/pub/telephony/asterisk/releases
```

```
/asterisk-13.7.2.tar.gz
wget http://downloads.asterisk.org/pub/telephony/libpri/releases
/libpri-1.4.15.tar.gz
wget http://downloads.asterisk.org/pub/telephony/asterisk/releases
/asterisk-addons-1.6.2.4.tar.gz
```

dekomprimace stažených souborů:

```
tar –xvzf asterisk-13.7.2.tar.gz && tar –xvzf lippri-1.4.15.tar.gz &&
tar –xvzf asterisk-addons-1.6.2.4.tar.gz
```

dále za použití níže uvedených příkazů dokončíme instalaci Asterisku:

```
make all
make install
make config
./configure
make menuselect
make config
```

Následně byly pro naše potřeby testování v konfiguračním souboru *sip.conf* nastaveny účty:

```
[2000];YateClient
type=peer
secret=1234
host=dynamic
dtmfmode=rfc2833
disallow=all
allow=g722
allow=ulaw
```

```
[2001];GRANDSTREAM
type=peer
secret=1234
host=dynamic
dtmfmode=rfc2833
disallow=all
allow=g722
allow=ulaw
```

4.3 Kali-linux

Na fyzickém PC byl nainstalován operační systém Kali-linux verze 2016-1, který obsahuje soubor penetračních nástrojů podporujících technologii CUDA, které budou použity při vlastním testování. Tento operační systém byl zvolen z důvodu jeho komplexnosti penetračních nástrojů z různých oblastí, včetně VoIP. Zbylé penetrační nástroje byly doinstalovány do tohoto operačního systému.

4.4 Suricata

IDS Suricata byl nainstalován na PC s konfigurací hardware uvedeným v tabulce 4.1 [31] [32]. Při konfiguraci byly použity dvě grafické karty NVIDIA GeForce 590 s CUDA, kdy při jejich zapojení byla použita technologie od firmy NVIDIA SLI (Scalable Link Interface). Jedná se o technologii, která zvyšuje výkon skoro o dvojnásobek oproti řešení s jednou grafickou kartou [33].

Tabulka 4.1 *Porovnání CPU a GPU u PC s CUDA.*

	Intel i7-4770k	**nVidia GeForce 590**
Frekvence [GHz]	3,9	1,7
Počet tranzistorů	$1{,}4*10^{12}$	$3*10^{12}$
Počet jader	8	1024
Propustnost [Gbit/s]	25,6	327,7
Výkon	177 Gflops	6 Tflops
Výrobní technologie [nm]	22	40
Maximální výkon [W]	84	365

Instalace IDS Suricata podporující technologii CUDA byla provedena kompilací zdrojových kódů za použití níže uvedených příkazů:

Ze stránek `http://www.nvidia.com/Download/index.aspx` si stáhneme ovladač grafické karty (pro nás GeForce 590) a cuda toolkit (v současné době – prosinec 2015 je ve verzi 7.5.18) následujícím příkazem:

```
wget http://developer.nvidia.com/cuda_7.5.18_linux.run
```

poté přidělíme oprávnění k instalaci:

```
chmod 655 cuda_7.5.18_linux.run
chmod 655 NVIDIA-Linux-x86_64-352.63.run
```

doinstalujeme potřebné balíčky:

```
sudo apt-get -y install libpcre3 libpcre3-dbg libpcre3-dev \
```

```
sudo apt-get -y install build-essential autoconf automake libtool
libpcap-dev libnet1-dev \
```

```
sudo apt-get -y install libyaml-0-2 libyaml-dev zlib1g zlib1g-dev
libcap-ng-dev libcap-ng0 make flex bison git
```

spustíme instalační balíček cuda toolkit a poté ovladač grafické karty:

```
sudo ./cuda_7.5.18_linux.run
```

```
sudo ./NVIDIA-Linux-x86_64-352.63.run
```

```
reboot
```

po restartu je třeba nainstalovat starší verzi sady překladačů gcc (GNU Compiler Collection) 4.4:

```
apt-get install gcc-4.4 gcc-4.4-base g++-4.4
```

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.5
40 --slave /usr/bin/g++ g++ /usr/bin/g++-4.8
```

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.4
60 --slave /usr/bin/g++ g++ /usr/bin/g++-4.4
```

```
sudo update-alternatives --config gcc
```

pro práci se vzdálenými repozitáři zadáme následující:

```
git clone git://phalanx.openinfosecfoundation.org/oisf.git
```

```
cd oisf/
```

```
git clone https://github.com/OISF/libhtp.git -b 0.5.x
```

```
./autogen.sh
```

```
./configure --enable-gccprotect --enable-profiling --enable-cuda \
```

```
--with-cuda-includes=/usr/local/cuda/include --with-cuda-
libraries=/usr/local/cuda/lib64/
```

Poté, co máme nainstalováno vše potřebné pro práci s CUDA si stáhneme IDS Suricata, kdy aktuální verze k 2.1.2016 je 2.0.11.

```
wget http://www.openinfosecfoundation.org/download/suricata-
2.0.11.tar.gz
```

```
gunzip suricata-2.0.11.tar.gz
```

```
tar –xvf suricata-2.0.11.tar
```

```
cd suricata-2.0.11/
```

```
./configure –-enable-gccprotect –-enable-profiling –-enable cuda --
with-cuda-includes=/usr/local/cuda/include --with-cuda-
libraries=usr/local/cuda/lib64/
```

```
make && make install-conf
```

dále stáhneme aktuální pravidla Suricaty:

```
./configure && make && make install-rules
```

Suricatu spustíme příkazem:

```
Suricata –c /etc/Suricata/Suricata.yaml
```

5 Předpoklady testování a samotná realizace měření

Ve výše uvedených kapitolách bylo uvedeno seznámení se s problematikou IP telefonie, kdy byla přiblížena oblast týkající se přenosu hlasu RTP pakety, signalizačního protokolu SIP a přenosového protokolu UDP. Dále byla představena oblast týkající se intrusivního detekčního nebo prevenčního systému a jeho různé umístění v síti, kdy poté byla uvedena instalace a konfigurace jednotlivých součástí VoIP s telefonní ústřednou Asterisk a IDS/IPS Suricata. Poté byly uvedeny penetrační nástroje mající za úkol otestovat správné nastavení IDS/IPS.

Nyní budou tyto informace využity v praktické části, která má za cíl otestovat výkonové zatížení IDS/IPS Suricata pomocí penetračních nástrojů. Při testování byla použita verze Suricaty 2.0.11.

5.1 První test - InviteFlood

Jako první bylo upraveno pro potřeby testování pravidlo zachytávání hrozeb aplikací InviteFlood, která má za cíl vyřadit z provozu telefonní ústřednu zasíláním neustálých SIP požadavků INVITE. Ve složce `/usr/local/etc/suricata/rules/` jsou definována pravidla pro VoIP telefonii. V tomto adresáři je již napsáno pravidlo proti záplavovému útoku InviteFlood v souboru `emerging-voip.rules`. Toto pravidlo bylo dále upraveno k potřebám testování.

Pravidlo definované jako obrana proti zasílání velkého množství zpráv INVITE za účelem vyčerpání hardwarových prostředků telefonní ústředny je zde:

```
alert udp $EXTERNAL_NET any → $HOME_NET 5060 (msg: "ET VOIP INVITE
Message Flood UDP"; content:"INVITE"; depth:6; threshold: type
both , track by_src, count 100, seconds 60;
reference:url,doc.emergingthreats.net/2009698; classtype:attempted-
dos; sid:2009698; rev:1;)
```

jednotlivé položky pravidla znamenají:

`Alert` – upozornění,
`udp` – upozornění se týká protokolu UDP,
`$` - znamená, že za tímto znakem následuje metoda definovaná v souboru `suricata.yaml`,
`EXTERNAL_NET` – vše mimo vnitřní síť (pravidlo bylo změněno, kde proměnná EXTERNAL_NET byla nahrazena proměnnou **`LOCAL_NET`** a v `suricata.yaml` byly definovány rozsahy IP adres `LOCAL_NET` pro potřeby našeho testování),
`any` – jakákoliv adresa z definovaného rozsahu,
→ směr datového toku, to znamená v našem případě z LOCAL_NET do HOME_NET,
`HOME_NET` - rozsahy IP adres definované v `suricata.yaml` domácí sítě s telefonní ústřednou,
5060 - port na kterém má být pravidlo aplikováno,
msg: na obrazovku se vypíše text zprávy v uvozovkách za slovem `msg:`,

content: "INVITE" - znamená, název metody v hlavičce metody INVITE protokolu SIP,
depth: 6 – znamená, kolik Byte od začátku hlavičky bude kontrolováno, v našem případě 6, jelikož název INVITE se skládá z šesti Byte,
treshold: type both, track by_src, count 100, second 60 – znamená, že bude v časovém limitu 60 sekund vygenerováno pouze jedno varování za splnění podmínky překročení limitu počtu 100 událostí,
reference:url,doc.emergingthreats.net/2009698 – odkaz na stránku týkající se tohoto pravidla, kde je možno poskytnout zpětnou vazbu tvůrcům tohoto pravidla a odkaz na dokument k tomuto pravidlu,
classtype:attempted-dos; sid:2009698; rev:1 – klasifikace pravidla podle definice v classification.config umístěného v /usr/local/etc/suricata/rules.

Příkaz pro InviteFlood byl v následující syntaxi:

inviteflood eth0 2000 192.168.0.1 192.168.0.3 1000000, kde:

eth0 je název sítě s telefonní ústřednou Asterisk,
2000 je název SIP účtu telefonní ústředny Asterisk, kdy v našem případě bude z tohoto účtu zasíláno níže uvedené množství zpráv Invite,
adresa **192.168.0.1** je adresou sítě,
adresa **192.168.0.3** je adresa telefonní ústředny,
1000000 je číslo udávající množství zaslaných zpráv Invite na telefonní ústřednu Asterisk.

Spuštění Suricaty proběhlo příkazem:

```
sudo/usr/local/bin/suricata -c /usr/local/etc/suricata/suricata.yaml
-i --init-errors-fatal
```

Z důvodu, že se jednalo o první testování, bylo v jeho průběhu změněno pravidlo, kdy místo upozornění alert bylo v pravidlu uvedeno varování warning nebo drop (současně s nastavením v suricata.yaml v položce host-mode: IPS), ale toto nemělo na výkonnost Suricaty žádný vliv.

Ze stejného důvodu prvního testování byl testován také rozdíl hardwarového zapojení grafických karet podporujících technologii CUDA s jednou grafickou kartou a s dvěma grafickými kartami spojených paralelně pomocí technologie SLI. Tímto byl zjištěn nestabilní výsledek testování oproti zapojení s jednou grafickou kartou, který se lišil při jednotlivých opakovaných měření i o 25%. Dále výsledek výkonového testování Suricaty v počtu zachycených paketů byl oproti zapojení s jednou grafickou kartou nižší o 5 až 30%. U opakovaných měření v zapojení s jednou grafickou kartou se výsledek lišil v rozmezí 0,1% až 2%. Suricata tedy nedokáže pracovat s více grafickými kartami natolik efektivně jako s jednou grafickou kartou a z tohoto důvodu bylo dále výkonové testování prováděno pouze s jednou grafickou kartou NVIDIA GeForce 590. Konkrétní výsledky výkonového testování jsou uvedeny v tabulce 5.1 a na obrázku 5.3.

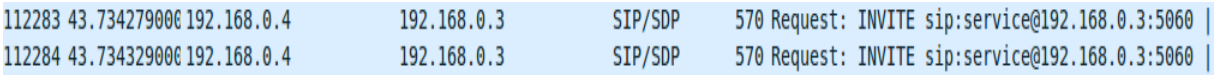
Po opakovaném testování byl zjištěn z logu stats.log, který je umístěn v

`/usr/local/var/log/suricata/` počet dekódovaných paketů na protokolu UDP 170.564/s až 173.625/s. Toto testování bylo několikrát opakováno a hodnota se po několika sekundách ustálila ve výše uvedených mezích.

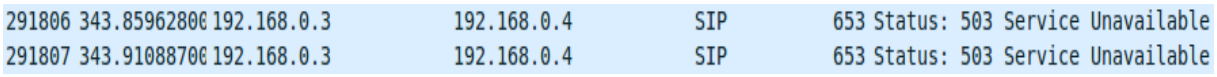
Testování bylo prováděno na síti s max. datovým přenosem 1Gb/s, to znamená, že teoretická max. propustnost sítě při velikosti paketu 587 Byte je 212.947 paketů. Suricata tedy pakety zkontrolovala a provoz na této síti nebyl nijak zvlášť omezen.

Pro kontrolu měření byl spuštěn paketový analyzátor Wireshark na PC, kde byl generován InviteFlood, čímž bylo zjištěno, že penetrační nástroj je schopen generovat v průměru 160000 p/s (paketů za sekundu). Suricata reagovala na spuštění penetračního nástroje nejdříve zachycením 95.456 p/s a po 11 sekundách se výsledek ustálil na výše uvedených hodnotách okolo 171.500 p/s.

Dále byl pro kontrolu spuštěn paketový analyzátor Wireshark na stroji s nainstalovanou telefonní ústřednou za účelem kontroly přijetí zpráv INVITE, obrázek 5-1. Což bylo ověřeno, jelikož pravidlo bylo upraveno jen jako varování a odeslané zprávy INVITE byly skutečně zaslány na stroj s telefonní ústřednou (po přijetí cca 120.000 paketů o velikosti každého 587 Byte byly systémové prostředky stroje s 4GB RAM a dvou-jádrovým CPU AMD Athlon s frekvencí 1,9GHz telefonní ústřednou vyčerpány, obrázek 5.2).



Obrázek 5.1 *Zachycené pakety INVITE.*



Obrázek 5.2 Zachycené pakety z Asterisku - nedostupná služba.

Při tomto testování bylo nastaveno v `suricata.yaml` `max-pending-packets: 60.000`, čímž bylo nakonfigurováno, že lze maximálně souběžně analyzovat 60.000 p/s.

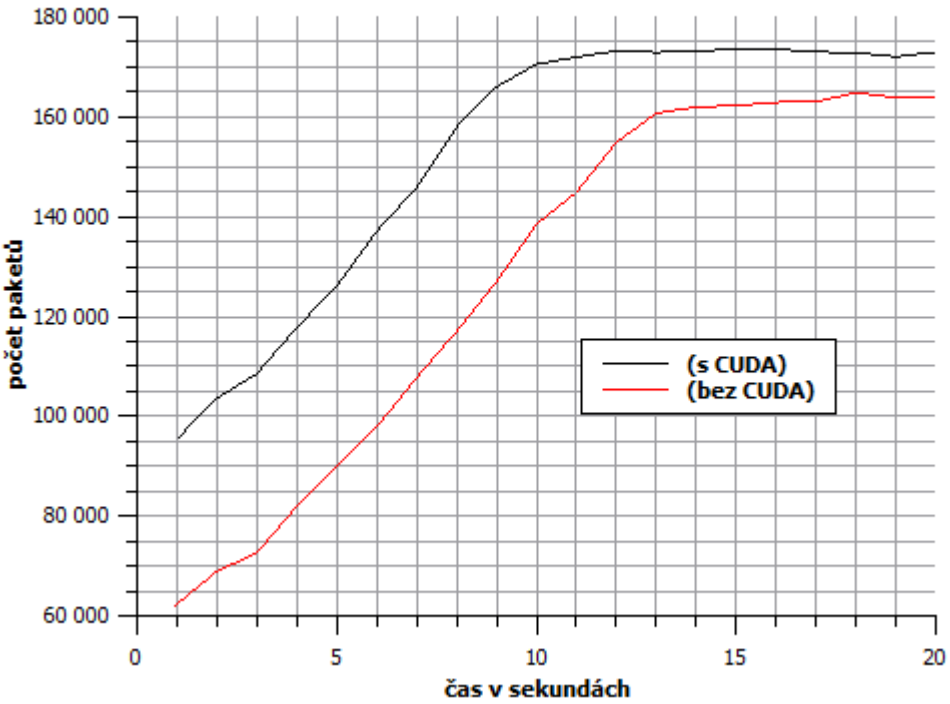
Stejné měření bylo opakováno po konfiguraci IDS Suricata bez použití výpočetního výkonu jader grafické karty (`--cuda enabled no`). Při tomto měření byly zjištěny podobné hodnoty, jako při konfiguraci s podporou CUDA s tím rozdílem, že se počet zpracovaných paketů po spuštění penetračního nástroje pohyboval při opakovaných měřeních okolo 62.000 p/s a počet dekódovaných paketů se po 15 sekundách ustálil na hodnotách v rozmezí 162.284 – 164.861 p/s.

Lze tedy tvrdit, že výkon Suricaty s konfigurací s podporou CUDA technologií je vyšší.

Dále bylo provedeno měření využití systémových prostředků stroje s nainstalovaným IDS Suricata pomocí aplikace htop, kdy bylo zjištěno, že zatížení procesoru se při konfiguraci s podporou CUDA pohybovalo v mezích 7,1-12,3% a při konfiguraci bez podpory CUDA v mezích 15,4-19,8%. Paměť RAM (4GB) byla při obou měřeních využita stejně, to znamená v maximálních hodnotách, které byly operačním systémem umožněny.

Tabulka 5.1 *Počet dekódovaných paketů nástroje InviteFlood s podporou CUDA a bez podpory CUDA.*

	Počet paketů za sekundu	
sekundy	s CUDA	bez CUDA
1	95456	62392
2	103639	68927
3	108556	72647
4	117826	81980
5	126082	89925
6	137097	97852
7	145871	107752
8	157982	117020
9	166026	126979
10	170564	138452
11	171982	144908
12	173223	154922
13	172952	160732
14	173145	161982
15	173625	162284
16	173582	162926
17	173022	163075
18	172865	164861
19	172062	163875
20	173018	164072



Obrázek 5.3 *Grafické porovnání počtu dekódovaných paketů penetračního nástroje InviteFlood.*

Dále bylo zjištěno, že IDS Suricata s podporou CUDA technologie nemá po 20 sekundách provozu žádné zpoždění detekce paketů v případě použití nástroje *InviteFlood,* což bylo zjištěno ze souboru `stats.log`, kdy bylo v konfiguračním souboru `suricata.yaml` nastaveno ukládání statistiky výkonu zpracování po 1 sekundě. V případě spuštění IDS Suricata bez podpory technologie CUDA došlo po 20-ti sekundách, ke 4 sekundovému zpoždění detekce paketů, kdy ale po 87 sekundách ke zpoždění detekce paketů již nedocházelo.

Pro zajímavost byl při jednotlivých testech zjišťován příkon stroje s nainstalovaným IDS Suricata v konfiguraci s podporou a bez podpory technologie CUDA komerčním měřičem příkonu BaseTech Cost Control 3000 CZ s max. odchylkou naměřené hodnoty 5% [57]. Tyto naměřené hodnoty byly zaokrouhleny na desítky. Příkon pracovního stroje s IDS/IPS Suricata byl v průběhu výkonnostního testování s konfigurací technologie CUDA 280W a bez konfigurace technologie CUDA 120W.

5.2 SIPvicious - Svcrack

Z této sady 5- ti nástrojů byl ve výkonovém testování využit nástroj *svcrack* pro jeho schopnost zasílat kontinuálně žádosti o registraci na PBX. *Svcrack* využívá slovníkový útok za účelem prolomení hesla k účtu klienta telefonní ústředny, v našem případě *Asterisku.*

Jako první bylo třeba vygenerovat slovník hesel, kdy za tímto účelem byl použit *crunch,* který byl použit v syntaxi uvedenou na obrázku 5.4.

```
crunch 7 7 abcdefghijklmnopqrstuvwxyz -o /root/Plocha/slovnik_suricata.txt
```

Obrázek 5.4 Příkaz crunch, kterým byl vygenerován slovník: slovnik_suricata.txt.

Syntaxe výše uvedeného příkazu znamená, že bude vygenerován slovník o délce každého slova 7 znaků (první číslo udává min. počet znaků, druhé číslo udává max. počet znaků). Jedná se o permutaci s opakováním znaků uvedených za druhou číslicí 7 [56]. Za parametrem `-o` je uvedena cesta, kde se má soubor s příponou `txt` uložit, v našem případě soubor `slovnik_suricata.txt`.

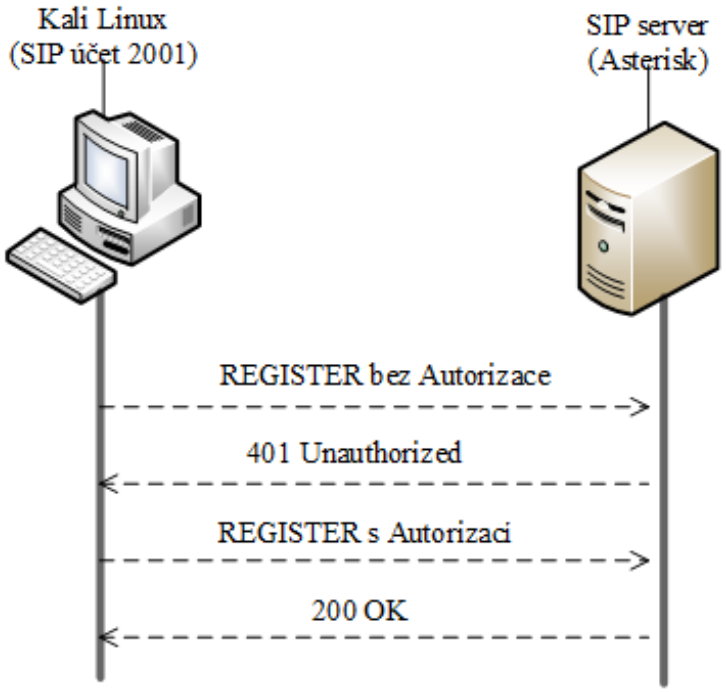
Dále byl příkazem uvedeným na obrázku 5.5 spuštěn nástroj *svcrack*. Vygenerovaný slovník příkazem *crunch* uvedeným výše má velikost 64,3 GB, což k testování výkonnosti Suricaty stačí (slovník samozřejmě neobsahuje heslo k testovanému účtu).

```
svcrack -u2001 -d  slovnik_suricata.txt 192.168.0.3
```

Obrázek 5.5 Syntaxe svcrack při spuštění testu výkonnosti Suricaty.

Parametr `-u2001` znamená název SIP účtu `2001` registrovaného u telefonní ústředny Asterisk, kterým se chceme zaregistrovat u Asterisku na IP adrese 192.168.0.3, viz. testovací topologie na obrázku 4.2.

Příkazem *svcrack* bylo generováno 11.086 paketů REGISTER s autorizací i bez autorizace za sekundu. Jednalo se o pakety REGISTER bez autorizace o velikosti 387 Byte a pakety REGISTER s autorizací o velikosti 576 Byte, viz obrázek 5.6, 5.7.



Obrázek 5.6 *Registrace SIP účtu na SIP server.*

```
1601 4…  192.168.0.4    192.168.0.3    SIP    387 Request: REGISTER sip:192.168.0.3  (1 binding)
1602 4…  192.168.0.3    192.168.0.4    SIP    551 Status: 401 Unauthorized |
1603 4…  192.168.0.4    192.168.0.3    SIP    576 Request: REGISTER sip:192.168.0.3  (1 binding)
1604 4…  192.168.0.3    192.168.0.4    SIP    508 Status: 403 Forbidden |
1605 4…  192.168.0.4    192.168.0.3    SIP    386 Request: REGISTER sip:192.168.0.3  (1 binding)
1606 4…  192.168.0.3    192.168.0.4    SIP    550 Status: 401 Unauthorized |
1607 4…  192.168.0.4    192.168.0.3    SIP    576 Request: REGISTER sip:192.168.0.3  (1 binding)
1608 4…  192.168.0.3    192.168.0.4    SIP    508 Status: 403 Forbidden |
1609 4…  192.168.0.4    192.168.0.3    SIP    385 Request: REGISTER sip:192.168.0.3  (1 binding)
1610 4…  192.168.0.3    192.168.0.4    SIP    549 Status: 401 Unauthorized |
1611 4…  192.168.0.4    192.168.0.3    SIP    574 Request: REGISTER sip:192.168.0.3  (1 binding)
1612 4…  192.168.0.3    192.168.0.4    SIP    506 Status: 403 Forbidden |
```

Obrázek 5.7 *Výpis z paketového analyzátoru REGISTER, 401 Unauthorized, 403 Forbidden.*

Dále byly generovány telefonní ústřednou Asterisk odpovědi 401 Unauthorized o velikosti 500 Byte a 403 Forbidden o velikosti 508 Byte. Celkem byl tedy IDS Suricata zatížen 22.172 p/s (11.086p/s x 2). Na obrázku 5.8 lze vidět generované poznámky telefonní ústřednou Asterisk o přijetí špatného hesla k danému SIP účtu.

```
[Mar 28 19:13:36] NOTICE[1159]: chan_sip.c:28003 handle_request_register: Regist
ration from '"2000" <sip:2000@192.168.0.3>' failed for '192.168.0.4:5060' - Wron
g password
[Mar 28 19:13:36] NOTICE[1159]: chan_sip.c:28003 handle_request_register: Regist
ration from '"2000" <sip:2000@192.168.0.3>' failed for '192.168.0.4:5060' - Wron
g password
[Mar 28 19:13:36] NOTICE[1159]: chan_sip.c:28003 handle_request_register: Regist
ration from '"2000" <sip:2000@192.168.0.3>' failed for '192.168.0.4:5060' - Wron
g password
[Mar 28 19:13:36] NOTICE[1159]: chan_sip.c:28003 handle_request_register: Regist
ration from '"2000" <sip:2000@192.168.0.3>' failed for '192.168.0.4:5060' - Wron
g password
[Mar 28 19:13:36] NOTICE[1159]: chan_sip.c:28003 handle_request_register: Regist
ration from '"2000" <sip:2000@192.168.0.3>' failed for '192.168.0.4:5060' - Wron
g password
[Mar 28 19:13:36] NOTICE[1159]: chan_sip.c:28003 handle_request_register: Regist
ration from '"2000" <sip:2000@192.168.0.3>' failed for '192.168.0.4:5060' - Wron
g password
```

Obrázek 5.8 *Generované poznámky telefonní ústřednou Asterisk o přijetí špatného hesla k SIP účtu.*

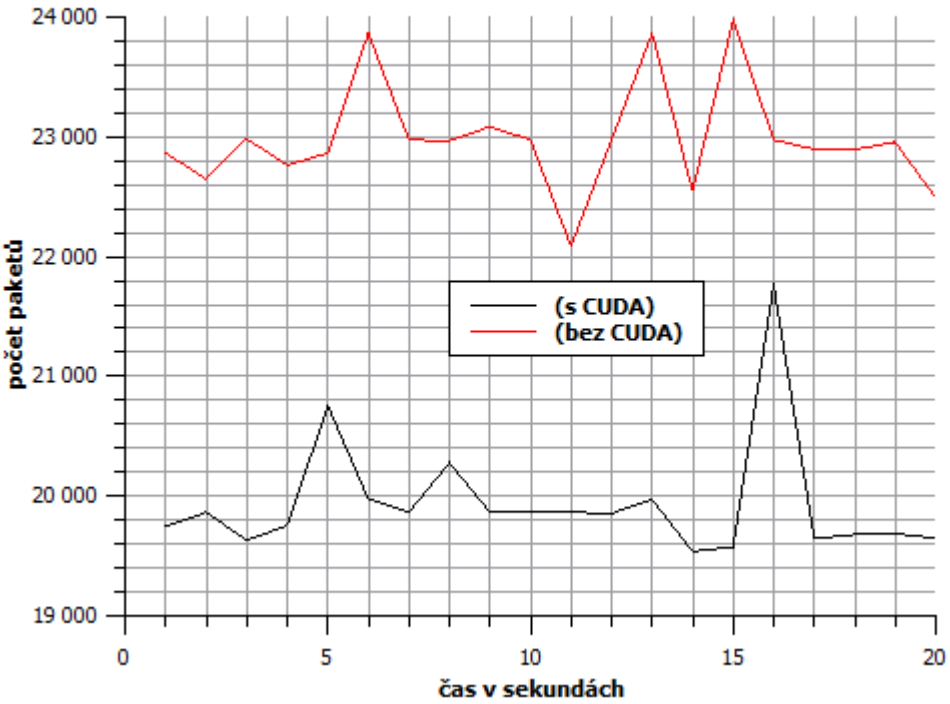
Při tomto testování bylo IDS Suricata s podporou technologie CUDA okamžitě detekováno 19.749 p/s a při dalším testování se výsledky nijak nelišily, viz tabulka 5.2. Při testování IDS bez podpory technologie CUDA byl okamžitý výsledek 22.866 p/s, což je v tomto případě prvního měření o 3.117 p/s více. Při testování, které bylo opět 20x opakováno byly zjištěny obdobné výsledky, to znamená, že bez podpory technologie CUDA je IDS Suricata v tomto případě výkonnější, viz obrázek 5.9.

Využití CPU počítače při výkonovém testování IDS Suricata se pohybovalo v rozmezí 1,3-5,4% a při testování IDS Suricata bez podpory technologie CUDA v rozmezí 10,5-16,1%.

Velikost využité paměti RAM IDS Suricata byl při konfiguraci s podporou technologie CUDA v rozmezí 1563-1625 MB a při konfiguraci bez podpory technologie CUDA v rozmezí 1823-2106 MB.

Tabulka 5.2 *Počet dekódovaných paketů nástroje Svcrack s podporou CUDA a bez podpory CUDA.*

	Počet paketů za sekundu	
sekundy	s CUDA	bez CUDA
1	19749	22866
2	19864	22648
3	19632	22984
4	19753	22764
5	20754	22865
6	19976	23865
7	19865	22975
8	20278	22966
9	19865	23086
10	19865	22975
11	19865	22086
12	19855	22976
13	19972	23865
14	19537	22548
15	19575	23975
16	21764	22975
17	19647	22896
18	19678	22896
19	19686	22955
20	19647	22489



Obrázek 5.9 *Grafické porovnání počtu dekódovaných paketů penetračního nástroje Svcrack.*

Při výkonovém testování IDS Suricata v konfiguraci s podporou technologie CUDA docházelo ke zpoždění části paketů do 1 sekundy. Toto bylo zjištěno ze souboru stats.log, kde bylo patrné, že po ukončení slovníkového útoku nedocházelo k další detekci paketů, jako v případě penetračního nástroje InviteFlood (detekce probíhala i 4s po ukončení útoku). Rovněž paměť RAM byla při výkonovém testování nástroje Svcrack méně vytížená oproti nástroji InviteFlood.

Při testování Svcrack stoupla využitá kapacita paměti RAM IDS Suricata z 103MB v některých případech až na 2106 MB, kdežto při testování InviteFlood stoupla kapacita paměti RAM využití IDS Suricata v některých případech dokonce až na 3663 MB z celkové použitelné paměti 3834 MB.

U tohoto testování se tedy projevilo, že pokud IDS Suricata pracuje střídavě i s odpověďmi z Asterisku, to znamená, že musí-li střídavě detekovat pakety ze dvou zdrojů provozu, je výkonnost IDS Suricata efektivnější bez konfigurace s podporou technologie CUDA a to o 15,7% oproti konfigurace s podporou technologie CUDA. Samozřejmě pokud se nebude brát v úvahu zvýšený výkon procesoru počítače z rozmezí 1,3-5,4% s podporou technologie CUDA oproti 10,5-16,1% bez konfigurace podpory technologie CUDA. A také se nebude brát ohled na zvýšené nároky na paměť RAM v rozmezí o 260-481MB. Příkon pracovního stroje s IDS/IPS Suricata byl v průběhu výkonnostního testování s konfigurací technologie CUDA 260W a bez konfigurace technologie CUDA 110W.

5.3 RTPinsertSound, RTPmixSound a RTPflood

Při testování těchto třech nástrojů bylo zjištěno, že jsou nevhodné k výkonovému testování IDS/IPS Suricata z důvodu, že k probíhajícímu telefonnímu hovoru přidávají RTP pakety (RTPinsertSound a RTPmixSound) nebo jsou generovány (RTPflood), čímž lze dosáhnout provozu zatížení sítě max. několika stovkami paketů za sekundu. Na tak malém provozu nebylo možno při testování rozpoznat rozdíl při konfiguraci IDS/IPS Suricata s podporou technologie CUDA a bez ní.

5.4 SIPp

Jak už bylo uvedeno výše, nejedná se přímo o penetrační nástroj, ale o aplikaci umožňující simulovat telefonní hovory na telefonní ústředně, v našem případě na Asterisku. Nástroj dokáže vygenerovat silný VoIP síťový provoz, náročný na detekci VoIP paketů, což je žádoucí pro výkonové testování IDS Suricata v konfiguraci s podporou technologie CUDA v porovnání s konfigurací IDS nepodporující tuto technologii.

Tímto nástrojem bylo provedeno několik druhů výkonového testování, kdy byly na telefonní ústřednu Asterisk zasílány žádosti INVITE na kterou telefonní ústředna reagovala odmítnutím žádosti, dále byla Suricata testována v situaci, kdy byly na telefonní ústřednu zasílány zprávy BYE. V poslední, scénáři byla výkonnost Suricaty testována na rychlost detekce paketů žádosti REGISTER a dále uměle zvětšenou žádost REGISTER, která musela být rozdělena z důvodu její velikosti do fragmentů, tudíž Suricata musela jednotlivé fragmenty žádosti REGISTER detekovat, poté spojit a zkontrolovat, zda pakety nevyhovují nějakému z definovaných pravidel.

5.4.1 SIPp a žádost INVITE

Při tomto testování příkazem `sipp uac -r 240.000 -rp 1000 192.168.0.3` bylo generováno vždy okolo 60.911 žádostí *INVITE* za 1 sekundu a stejný počet odmítnutí žádosti telefonním ústřednou *403 Forbidden*, viz obrázek 5.10. To znamená, že v testovací topologii IDS/IPS Suricata zachytil dvojnásobek z počtu generovaných paketů za 1 sekundu.

9682…	55.778307160	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.778310378	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.778311940	192.168.0.3	192.168.0.4	SIP	503 Status: 403 Forbidden |
9682…	55.778316385	192.168.0.3	192.168.0.4	SIP	503 Status: 403 Forbidden |
9682…	55.778318158	192.168.0.3	192.168.0.4	SIP	500 Status: 403 Forbidden |
9682…	55.778319934	192.168.0.3	192.168.0.4	SIP	500 Status: 403 Forbidden |
9682…	55.778321875	192.168.0.3	192.168.0.4	SIP	503 Status: 403 Forbidden |
9682…	55.778378351	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.778406040	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.778432272	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.778454760	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.778485529	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.778502871	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.778517822	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.778716865	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.778722073	192.168.0.3	192.168.0.4	SIP	503 Status: 403 Forbidden |
9682…	55.778726333	192.168.0.3	192.168.0.4	SIP	500 Status: 403 Forbidden |
9682…	55.778728426	192.168.0.3	192.168.0.4	SIP	503 Status: 403 Forbidden |
9682…	55.778730121	192.168.0.3	192.168.0.4	SIP	503 Status: 403 Forbidden |
9682…	55.778731885	192.168.0.3	192.168.0.4	SIP	503 Status: 403 Forbidden |
9682…	55.778733820	192.168.0.3	192.168.0.4	SIP	503 Status: 403 Forbidden |
9682…	55.779150334	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060
9682…	55.779154801	192.168.0.4	192.168.0.3	SIP/SDP	567 Request: INVITE sip:service@192.168.0.3:5060

Obrázek 5.10 Zachycené pakety žádosti INVITE a odpovědi 403 Forbidden.

Tento nástroj oproti penetračnímu nástroji InviteFlood se snaží sestavit určitý počet spojení definovaného za parametrem `-r`. V našem případě jsem úmyslně nastavil hodnotu 240.000, která je nad hodnotou limitu max. propustnosti sítě testované topologie při velikosti paketu 567 Byte obsahující žádost *INVITE*. Za dalším parametrem `-rp` se uvádí, za jak dlouhou dobu má dojít k sestavení počtu spojení, kdy v našem případě hodnota 1000 znamená, že k sestavení 240.000 spojení má dojít za 1000 ms. Dále IP adresa 192.168.0.3 je IP adresou telefonní ústředny Asterisk na které se mají hovory sestavit, viz obrázek testovací topologie 4.2 .

To znamená, že se SIPp skutečně snaží sestavit reálné spojení oproti nástroji InviteFlood, který jen zasílal neustále žádosti INVITE, dokud nedošlo k nedostupnosti služby telefonní ústředny. Z pohledu zatížení testovací topologie a tím i IDS Suricata by se toto testování dalo přirovnat předešlému testování v případě slovníkového útoku z důvodu, že IDS Suricata detekuje jednak pakety žádosti *INVITE* z Kali-Linux a jednak detekuje pakety *403 Forbiden* z telefonní ústředny Asterisk o stejném počtu.

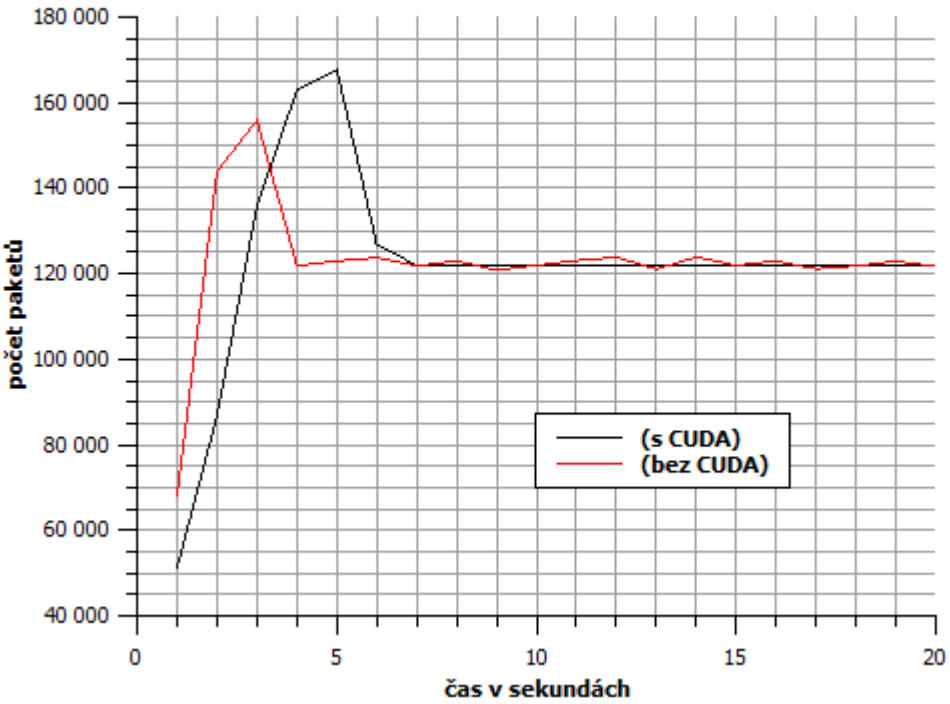
Při tomto testování bylo IDS Suricata s podporou technologie CUDA okamžitě detekováno 51.235 p/s a dále 86.752, 135.865, 162.875, 167.754, 126.865, a poté již stabilně 121.862-122.074 p/s, viz tabulka 5.3. Při testování IDS bez podpory technologie CUDA byl okamžitý výsledek 67.972 p/s a dále 143.873, 155.853, 120.853-123.861 p/s. Při testování, které bylo pro objektivnost 20x opakováno bylo zjištěno, že IDS Suricata dokáže detekovat pakety IP telefonie s konfigurací technologie CUDA i bez ní. Při konfiguraci bez podpory technologie CUDA se o 3 sekundy rychleji dokáže IDS Suricata přizpůsobit zátěži sítě, to znamená o 42% rychleji.

Využití CPU počítače při výkonovém testování IDS Suricata se pohybovalo v rozmezí 1,6-6,7% a při testování IDS Suricata bez podpory technologie CUDA v rozmezí 12,5-24,2%.

Velikost využité paměti RAM IDS Suricata byl při konfiguraci s podporou technologie CUDA v rozmezí 1863-2235 MB a při konfiguraci bez podpory technologie CUDA v rozmezí 2185-2582 MB.

Tabulka 5.3 *Počet dekódovaných paketů nástroje SIPp generujícího žádost INVITE s podporou CUDA a bez podpory CUDA.*

	Počet paketů za sekundu	
Sekundy	s CUDA	bez CUDA
1	51235	67972
2	86752	143873
3	135865	155853
4	162875	121874
5	167754	122962
6	126865	123732
7	121964	121852
8	121885	122952
9	122006	120853
10	121925	121962
11	121862	123072
12	122043	123841
13	121892	121062
14	121885	123861
15	122074	121951
16	122011	122951
17	121869	121139
18	122042	121831
19	121959	122927
20	122037	121742



Obrázek 5.11 *Grafické porovnání počtu dekódovaných paketů aplikace SIPp generující žádost INVITE.*

Při výkonovém testování IDS Suricata v konfiguraci s podporou technologie CUDA docházelo po startu aplikace SIPp generující žádost INVITE ke zpoždění detekce části paketů, ale po 7 sekundách provozu již byla detekce paketů bez zpoždění. Při konfiguraci IDS Suricata bez podpory technologie CUDA docházelo rovněž po generování zátěže ke zpoždění detekce části paketů, ale po 4 sekundách od startu aplikace SIPp již ke zpoždění detekce nedocházelo. Toto bylo opět zjištěno ze souboru stats.log.

U tohoto testování se opět projevilo, že pokud IDS Suricata detekuje síťový provoz z více zdrojů, pracuje efektivněji bez konfigurace s podporou technologie CUDA, což se v tomto případě projevilo rychlejším přizpůsobením se síťovému VoIP provozu, a to o 42% oproti konfiguraci s podporou technologie CUDA. Rovněž je ale tato "akcelerace" na úkor zvýšeného výkonu procesoru počítače z rozmezí 1,6-6,7% s podporou technologie CUDA oproti 12,5-24,2% bez konfigurace podpory technologie CUDA a dále na úkor zvýšeného nároku na paměť RAM v rozmezí o 332-347 MB. Příkon pracovního stroje s IDS/IPS Suricata byl v průběhu výkonnostního testování s konfigurací technologie CUDA 270W a bez konfigurace technologie CUDA 120W.

Zvýšené nároky na systémové prostředky využívající IDS Suricata jsou ale trvalé při detekci silného síťového provozu, proto je třeba brát v úvahu, v jakém síťovém provozu bude IDS Suricata nasazen a na jak výkonném pracovním stroji.

Jelikož se na první pohled nezdá, že by naměřené hodnoty v rozmezí 7 -20 sekund byly nějak rozdílné, viz obrázek 5.11, byl proveden test na zjištění, zda si jsou naměřená data podobná či nikoliv. Nejdříve byl u naměřených hodnot od 7 do 20 sekund proveden test normality, viz obrázek 5.12. Tento test byl

proveden Shapiro-Wilk testem na hladině významnosti 95%, kdy bylo zjištěno, že data naměřených hodnot detekce paketů IDS Suricata v konfiguraci s technologií CUDA i bez ní pochází z normálního rozdělení, viz obrázek 5.12.

[6.4.2016 19:35:39 "Table1"] Normality Test (Shapiro - Wilk)

Dataset	N	W	P Value	Decision
Table1_2	14	0,9250136801566	0,2594214253179	Normal at 0,05 level

[6.4.2016 19:35:50 "Table1"] Normality Test (Shapiro - Wilk)

Dataset	N	W	P Value	Decision
Table1_1	14	0,9060197043211	0,1379389237	Normal at 0,05 level

Obrázek 5.12 *Test normality Shapiro - Wilk testem.*

Poté byl proveden dvouhodnotový Studentův t-test za účelem zjištění statisticky významných rozdílů, kdy byl potvrzen předpoklad, jak je patrné z grafu, že naměřená data si jsou podobná, viz obrázek 5.13.

Alpha	Sample Size	Power
0,05	14	0,209757301269 (actual)

[6.4.2016 19:46:14 "Table1"] Two Sample Independent t-Test

Sample	N	Mean	Standard Deviation	Variance	Standard Error
Table1_1	14	121 961	74,54580419498	5 557,076923077	19,92320420851
Table1_2	14	122 285,4285714	984,0612640626	968 376,5714286	263,0014355513

Difference of Means: -324,4285714286

Null Hypothesis: Mean1 - Mean2 = 0
Alternative Hypothesis: Mean1 - Mean2 <> 0

t	DoF	P Value
-1,230037712061	26	0,2296970095795

At the 0,05 level, the difference of the population means is not significantly different than the test difference (0).

Obrázek 5.13 *Dvouhodnotový Studentův t-test.*

5.4.2 SIPp a odpověď BYE

Při tomto testování příkazem `sipp uac -r 240.000 -rp 1000 192.168.0.3 -default_behaviors -bye` byly generovány žádosti *BYE* na které telefonní ústředna Asterisk reagovala odpověďmi *404 Not Found,* viz obrázek 5.14.

80035	51.952068118	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80036	51.956137424	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80037	51.961233687	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80038	51.965338885	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80039	51.970405598	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80040	51.974487225	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80041	51.978563896	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80042	51.983648760	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80043	51.987753778	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80044	51.991818017	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80045	51.995901122	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80046	52.000982976	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found
80047	52.005071160	192.168.0.3	192.168.0.4	SIP	497 Status: 404 Not Found

Obrázek 5.14 Zachycené pakety odpovědi 404 Not Found telefonní ústředny Asterisk.

Aplikací SIPp bylo generováno 25.314 žádostí *BYE* za 1 sekundu a stejný počet odpovědí telefonní ústředny Asterisk, to znamená, že byl ze dvou portů přepínače generován VoIP provoz o celkem 50.628 p/s, který musel být IDS Suricata detekovaný při spuštění aplikace SIPp v syntaxi `-default_behaviors -bye`. Na obrázku 5.14 je IP adresa 192.168.0.3 adresou telefonní ústředny Asterisk a IP adresa 192.168.0.4 adresou zdroje žádostí *BYE*, kterým je generátor VoIP provozu SIPp spuštěný v operačním systému Kali-Linux, viz obrázek testovací topologie 4.2.

Toto testování lze přirovnat generování žádostí *INVITE* aplikací SIPp s tím rozdílem, že je výkon generátoru VoIP síťového provozu nižší a to o 35.597 p/s ze strany generátoru aplikace SIPp (z IP adresy 192.168.0.4). To znamená, že celkový provoz v testovací topologii klesl z 121.822 p/s na 50.628 p/s.

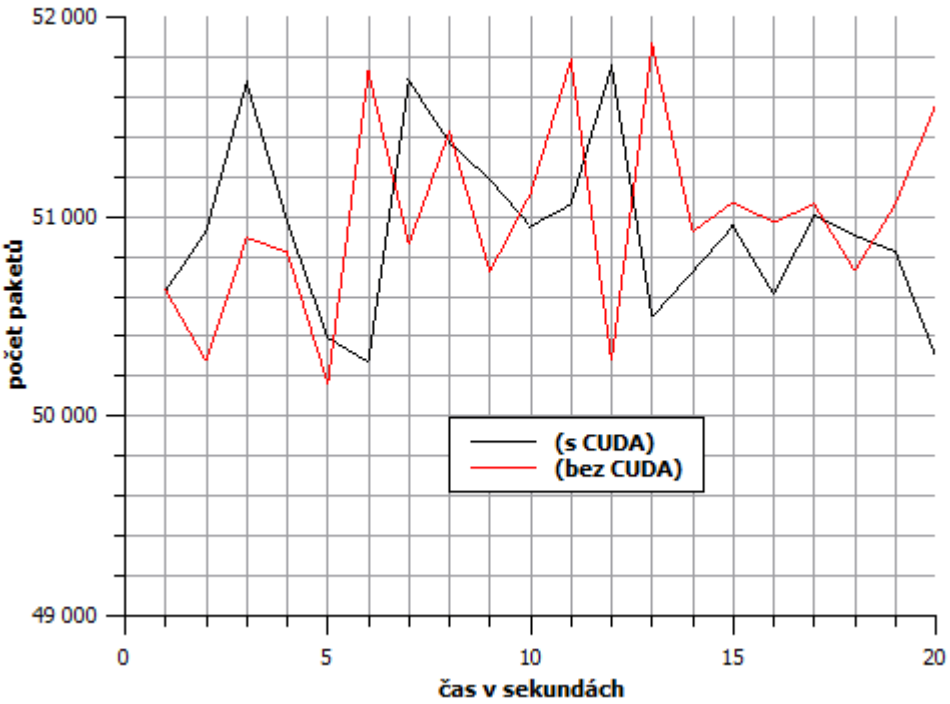
Při tomto testování bylo IDS Suricata s podporou technologie CUDA okamžitě detekován síťový VoIP provoz o velikosti 50.628 p/s a stejně tak i dále, to znamená bez zpoždění detekce paketů z obou portů přepínače. Síťový VoP provoz detekovaný IDS Suricata se pohyboval v rozmezí 50.274 – 51.752 p/s. Při testování IDS bez podpory technologie CUDA byl okamžitý výsledek 50.630 p/s a dále bez zpoždění detekce paketů stejně jako v konfiguraci s podporou technologie CUDA. Toto testování bylo pro objektivnost několikrát opakováno po dobu min. 20 sekund, kdy se síťový provoz pohyboval v rozmezí 50.162 – 51.862 p/s.

Využití CPU počítače při výkonovém testování IDS Suricata se pohybovalo v rozmezí 2,4-6,3% a při testování IDS Suricata bez podpory technologie CUDA v rozmezí 10,7-20,1%.

Velikost využité paměti RAM IDS Suricata byl při konfiguraci s podporou technologie CUDA v rozmezí 1625-1652 MB a při konfiguraci bez podpory technologie CUDA v rozmezí 1983-2065 MB.

Tabulka 5.4 *Počet dekódovaných paketů nástroje SIPp generujícího odpovědi BYE s podporou CUDA a bez podpory CUDA.*

	Počet paketů za sekundu	
sekundy	s CUDA	bez CUDA
1	50628	50630
2	50928	50276
3	51672	50895
4	50973	50822
5	50392	50162
6	50272	51725
7	51682	50862
8	51371	51427
9	51186	50724
10	50952	51116
11	51062	51784
12	51752	50276
13	50497	51862
14	50724	50925
15	50952	51072
16	50611	50972
17	51008	51062
18	50906	50728
19	50826	51062
20	50299	51563



Obrázek 5.15 *Grafické porovnání počtu dekódovaných paketů aplikace SIPp generující odpověď BYE.*

Při výkonovém testování IDS Suricata nedocházelo v konfiguraci s podporou technologie CUDA ani v konfiguraci bez této podpory ke zpoždění, což bylo zjištěno ze souboru stats.log, viz obrázek 5.16.

```
decoder.udp                | RxPcapem11        | 51186
```

Obrázek 5.16 *Ukázka z výpisu stats.log - počet dekódovaných paketů.*

Toto testování zatěžovalo síťový provoz ze všech dosud provedených testů nejméně. IDS Suricata detekovala síťový provoz ze dvou portů přepínače. U provedených výkonových testů nedocházelo při detekci paketů IDS Suricata ke zpoždění a výsledky tohoto testování detekce jsou téměř shodné a z tohoto důvodu se u testu "čistě" projevilo je využití systémových prostředků IDS Suricata v konfiguraci s podporou a bez podpory technologie CUDA.

Jak bylo uvedeno výše, využití CPU počítače se pohybovalo v rozmezí 2,4-6,3% a při testování IDS Suricata bez podpory technologie CUDA v rozmezí 10,7-20,1% a dále velikost využité paměti RAM byla při konfiguraci s podporou technologie CUDA v rozmezí 1625-1652 MB a při konfiguraci bez podpory technologie CUDA v rozmezí 1983-2065 MB. Rozdíl mezi oběma konfiguracemi ve využití CPU je 3,8-6,8% a ve využití paměti RAM je 358-413 MB. Opět se tedy u tohoto testování projevilo, že konfigurace IDS/IPS Suricata s podporou technologie CUDA méně vyčerpává systémové prostředky. Příkon pracovního stroje s IDS/IPS Suricata byl v průběhu výkonnostního testování s konfigurací technologie CUDA 250W a bez konfigurace technologie CUDA 90W.

Jelikož se opět naměřené hodnoty nezdály být statisticky významně rozdílné mezi oběma měřeními, viz obrázek 5.15, byl nejdříve proveden Shapiro-Wilkov test po ověření normality dat, viz obrázek 5.17,

kterým bylo zjištěno, že naměřená data pochází z normálního rozdělení a poté Studentův dvouhodnotový t-test.

[6.4.2016 19:53:41 "Table1"] Normality Test (Shapiro - Wilk)

Dataset	N	W	P Value	Decision
Table1_1	20	0,9435926096884	0,2800901782182	Normal at 0,05 level

[6.4.2016 19:53:53 "Table1"] Normality Test (Shapiro - Wilk)

Dataset	N	W	P Value	Decision
Table1_2	20	0,9513659098533	0,388300881963	Normal at 0,05 level

Obrázek 5.17 *Shapiro-Wilkov test normality naměřených dat.*

Studentovým dvouhodnotovým t-testem bylo zjištěno, že rozdíl naměřených hodnot detekce paketů IDS Suricata s konfigurací podpory technologie CUDA a bez podpory technologie CUDA není statisticky rozdílný na hladině významnosti 99%, viz obrázek 5.18.

At the 0,01 level, the difference of the population means is not significantly different than the test difference (0).

Obrázek 5.18 *Věta říkající, že rozdíl hodnot dvou měření není na hladině pravděpodobnosti 99% statisticky významný.*

5.4.3 SIPp a žádost REGISTER

Při tomto testování příkazem `sipp -s 1000 -ap 1000 -r 240000 192.168.0.3:5060 -sf register.xml` bylo generováno 42.865 žádostí *REGISTER* za sekundu bez čekání na odpověď telefonní ústředny Asterisk, viz obrázek 5.19. Dalo by se říct, že se v principu jedná o podobné chování penetračního nástroje *InviteFlood* s tím rozdílem, že jsou rychle za sebou generovány žádosti REGISTER bez čekání na odpověď telefonní ústředny. Velikost paketu *REGISTER* je 309 Byte oproti velikosti paketu *INVITE*, který byl 567 Byte. Parametr `-s 1000` udává počet uživatelů žádajících o registraci, parametr `-ap 1000` udává počet hesel k počtu uživatelů, parametr `192.168.0.3:5060` udává IP adresu telefonní ústředny a číslo portu na kterém bude komunikace probíhat, parametr `-sf register.xml` udává název souboru, kde je definována žádost REGISTER.

46012	1.733266000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46013	1.733292000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46014	1.733321000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46015	1.733346000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46016	1.733373000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46017	1.733398000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46018	1.733425000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46019	1.733452000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46020	1.733478000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46021	1.733505000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46022	1.733532000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46023	1.733558000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46024	1.733584000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46025	1.733612000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46026	1.733637000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46027	1.733665000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46028	1.733691000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3
46029	1.733717000	192.168.0.4	192.168.0.3	SIP	308 Request: REGISTER sip:192.168.0.3

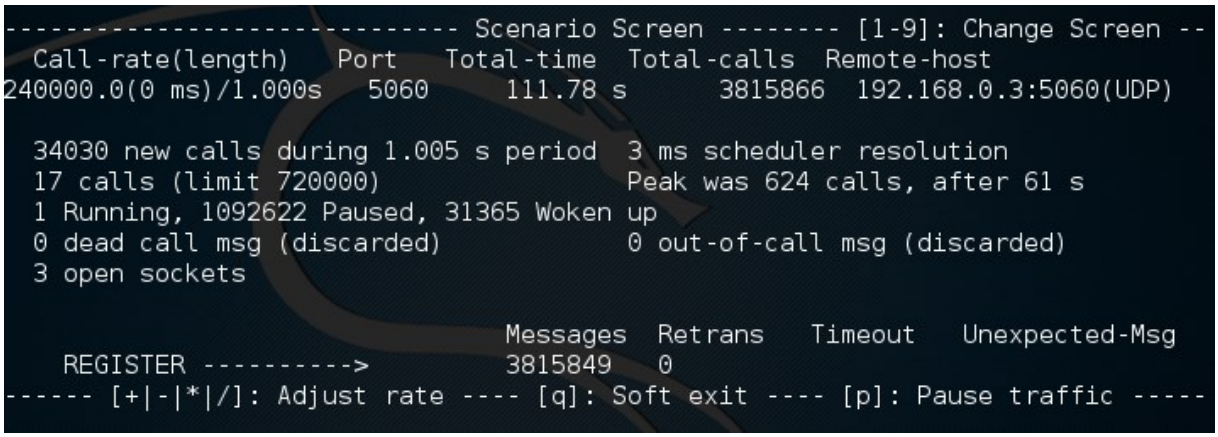
Obrázek 5.19 Zachycené pakety žádosti REGISTER generované aplikací SIPp.

Definici žádosti REGISTER zasílaná na telefonní ústřednu Asterisk jsem uložil do souboru *register.xml*. Syntaxe žádosti vypadá následovně:

```
1.<?xml version="1.0" encoding="ISO-8859-1" ?>
2.<!DOCTYPE scenario SYSTEM "register.xml">
3.<scenario name="REGISTER">
4.  <send>
5.    <![CDATA[
6.      REGISTER sip:[remote_ip] SIP/2.0
7.      Via: SIP/2.0/[transport] [local_ip]:[local_port]
8.      To: <sip:[service]@sip.com:[remote_port]>
9.      From: <sip:[service]@[remote_ip]:[remote_port]>
10.      Contact:
11.<sip:[service]@[local_ip]:[local_port]>;transport=[transport]
12.      Expires: 300
13.      Call-ID: [call_id]
14.      CSeq: 2 REGISTER
15.      Content-Length: 0
16. ]]>
17.  </send>
```

Výše uvedený scénář, stejně jako i jiné scénáře mají obvyklou strukturu jazyka XML. Řádky 1. a 2. uvozují, že se jedná o XML soubor s daným typem kódování a uloženém v souboru `register.xml`. Na řádku 3. je uvedeno jméno scénáře. Na řádcích 4. a 17. jsou tagy s logickým pojmenováním `send` pro zprávy, které mají být odeslány (dále například tag: `recv` vymezuje zprávu, kterou nástroj SIPp očekává). Samotné zprávy jsou ve scénáři začínající frází `CDATA`. Pole obsahující adresu SIP serveru je `remote_ip`. Pole `transport` udává typ transportního protokolu, `local_ip` a `local_port` udávají lokální adresu a port odesílatele, `call_id` je generováno nástrojem SIPp.

Na obr. 5.20 lze vidět hodnoty, které zobrazuje nástroj SIPp v průběhu jeho spuštění, v tomto případě při zasílání žádostí *REGISTER* na telefonní ústřednu Asterisk.



Obrázek 5.20 *Ukázka hodnot zobrazující nástroj SIPp v průběhu jeho spuštění.*

Nástroj SIPp se v tomto scénáři nesnaží sestavit reálné spojení, nýbrž zahlcuje síť, na které je provozována telefonní ústředna rychle se opakujícími žádostmi REGISTER. U tohoto nástroje je test podobný zasílání žádostí INVITE nástrojem SIPp s tím rozdílem, že telefonní ústředna na tyto žádosti nereaguje odesíláním odpovědí. Tento test byl proveden jednak z důvodu porovnání, když se opakuje obdobný scénář v případě zasílání INVITE s odpovědí nástrojem SIPp a jednak z důvodu, jak se ukáže, že nemá vliv na výsledek zátěžového testu detekce IDS Suricata o jaký typ žádosti se jedná, ale že má vliv jaké množství žádostí je IDS Suricata detekováno. To znamená, že v případě detekce žádostí INVITE s odpovědí telefonní ústředny Asterisk jsou nároky na IDS Suricata vyšší z důvodu detekce provozu z více portů a dále, že nároky na systémové prostředky při detekci provozu z jednoho portu jsou v tomto případě nižší než u detekce provozu z jednoho portu nástrojem InviteFlood, který dokázal generovat i 164.861 p/s. A jako poslední závislost na systémových prostředcích se projevilo, že IDS Suricata má v tomto případě nižší nároky při nižším provozu generovaného z jednoho portu v porovnání zasílání odpovědí BYE nástrojem SIPp, který v případě zasílání odpovědí BYE bylo generováno z portu nástroje SIPp a portu PBX Asterisk celkem generováno 50.630 p/s oproti 42.865 p/s (z jednoho portu).

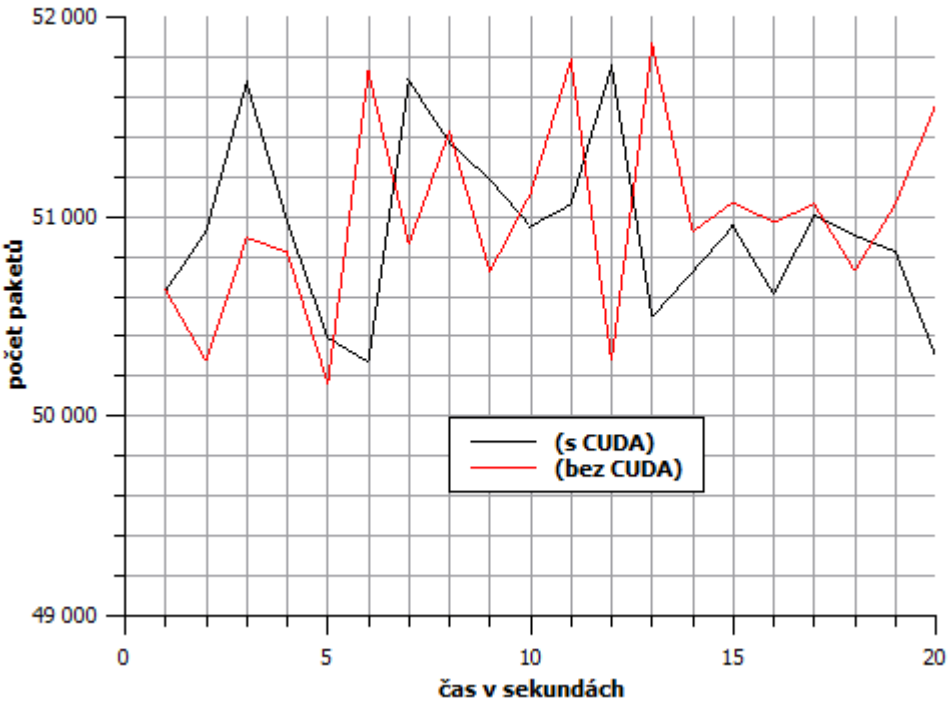
Při tomto testování bylo IDS Suricata s podporou technologie CUDA okamžitě detekováno 42.865 p/s a při dalším testování se výsledky nijak nelišily, viz tabulka 5.5. Při testování IDS bez podpory technologie CUDA byl okamžitý výsledek 42.325 p/s. Při testování, které bylo 20x opakováno, bylo zjištěno, že systémové nároky na IDS jsou v konfiguraci bez podpory technologie CUDA vyšší.

Využití CPU počítače při výkonovém testování IDS Suricata v konfiguraci s podporou technologie CUDA se pohybovalo v rozmezí 1,3-5,6% a při konfiguraci bez podpory technologie CUDA v rozmezí 11-18,6%.

Velikost využité paměti RAM IDS Suricata byl při konfiguraci s podporou technologie CUDA v rozmezí 1568-1682 MB a při konfiguraci bez podpory technologie CUDA v rozmezí 1853-1906 MB.

Tabulka 5.5 *Počet dekódovaných paketů nástroje SIPp generujícího žádost REGISTER s podporou CUDA a bez podpory CUDA.*

	Počet paketů za sekundu	
sekundy	s CUDA	bez CUDA
1	42865	42325
2	42863	42862
3	42962	42972
4	43062	42928
5	42962	42087
6	43962	42142
7	42963	43146
8	43932	43780
9	43427	43972
10	42925	42872
11	42978	42079
12	43325	43992
13	42036	43072
14	43675	43087
15	43678	43671
16	43895	42119
17	42187	43992
18	42081	42982
19	43080	43076
20	43907	42576



Obrázek 5.21 *Grafické porovnání počtu dekódovaných paketů aplikace SIPp generující žádost REGISTER.*

Při tomto testování nedocházelo ke zpožděné detekci paketů. Při testování byl rozdíl ve využití paměti RAM o 285 MB vyšší při konfiguraci IDS Suricata bez podpory technologie CUDA a dále využití CPU o 6,9 – 9,4%. Příkon pracovního stroje s IDS/IPS Suricata byl v průběhu výkonnostního testování s konfigurací technologie CUDA 250W a bez konfigurace technologie CUDA 70W.

Naměřené hodnoty detekovaných paketů IDS Suricata s konfigurací podpory technologie CUDA a bez podpory CUDA opět nebyly na první pohled rozdílné, viz obrázek 5.21, a proto byl u nich proveden test, zda jsou naměřená data statisticky rozdílná. Po provedeném ověření Shapiro-Wilkov testem, že data pochází z normálního rozdělení a naměřené hodnoty neobsahují odlehlá pozorování nebo chyby byl proveden Studentův t-test na hladině významnosti 95%, kdy nulová hypotéza byla nastavena tak, že byl předpoklad, že se data ve sloupcích rovnají. Tímto testem bylo zjištěno, že na hladině významnosti 95% nejsou naměřené hodnoty detekovaných paketů IDS Suricata významně rozdílné mezi naměřenými hodnotami konfigurace IDS Suricata s podporou technologie CUDA a bez podpory této technologie, viz poslední věta na obrázku 5.22.

[6.4.2016 20:15:09 "Table1"] Normality Test (Shapiro - Wilk)

Dataset	N	W	P Value	Decision
Table1_1	20	0,912572707348	0,07136444509414	Normal at 0,05 level
Table1_2	20	0,9116830624398	0,06862525657888	Normal at 0,05 level

[6.4.2016 20:16:04 "Table1"] Two Sample Independent t-Test

Sample	N	Mean	Standard Deviation	Variance	Standard Error
Table1_1	20	43 138,25	595,2077810577	354 272,3026316	133,0925059182
Table1_2	20	42 986,6	643,9623558792	414 687,5157895	143,9943602697

Difference of Means: 151,65

Null Hypothesis: Mean1 - Mean2 = 0
Alternative Hypothesis: Mean1 - Mean2 <> 0

t	DoF	P Value
0,7734025313604	38	0,4440689659135

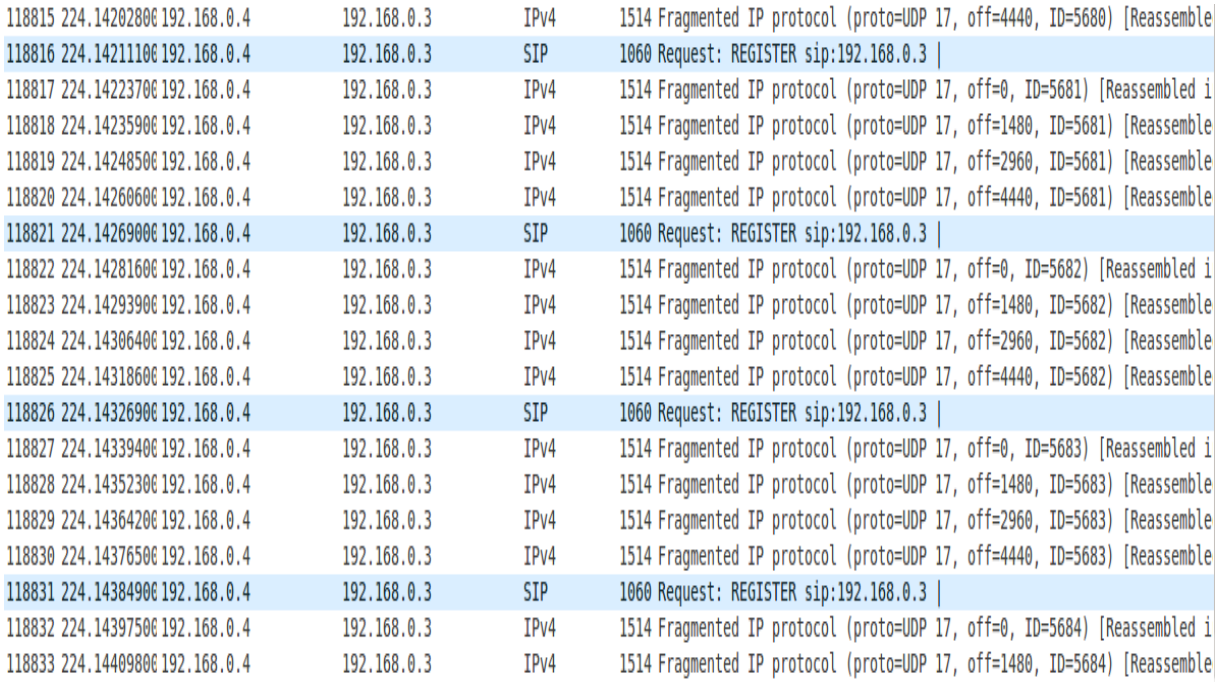
At the 0,05 level, the difference of the population means is not significantly different than the test difference (0).

Obrázek 5.22 *Log výsledku provedeného testu významnosti rozdílů naměřených hodnot.*

5.4.4 SIPp a fragmentovaná žádost REGISTER

Při tomto testování rovněž příkazem `sipp -s 1000 -ap 1000 -r 240000 192.168.0.3:5060 -sf register.xml` bylo generováno 15.611 paketů s žádostí *REGISTER* za sekundu bez čekání na odpověď telefonní ústředny Asterisk s tím rozdílem, že velikost žádosti *REGISTER* nebyla jako v předchozím případě 309 Byte, ale 7116 Byte, kterou bylo nutno rozdělit do

5 fragmentů o velikosti 4x1514 Byte a 1x1060 Byte, viz obrázek 5,23. Celkem bylo tedy generováno 78.055 p/s. Takováto žádost *REGISTER* byla vytvořena úmyslně, i když je dle mého názoru v praxi nereálná, ale chtěl jsem vyzkoušet, jak si IDS Suricata poradí v IP telefonii s fragmentací paketů při výkonovém testování v konfiguraci s podporou technologie CUDA a bez ní. Význam jednotlivých parametrů spouštěcího příkazu aplikace SIPp je samozřejmě stejný jako v předešlém testování.



118815	224.1420280	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=4440, ID=5680) [Reassemble
118816	224.1421110	192.168.0.4	192.168.0.3	SIP	1060 Request: REGISTER sip:192.168.0.3 |
118817	224.1422370	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=5681) [Reassembled i
118818	224.1423590	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=5681) [Reassemble
118819	224.1424850	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=2960, ID=5681) [Reassemble
118820	224.1426060	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=4440, ID=5681) [Reassemble
118821	224.1426900	192.168.0.4	192.168.0.3	SIP	1060 Request: REGISTER sip:192.168.0.3 |
118822	224.1428160	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=5682) [Reassembled i
118823	224.1429390	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=5682) [Reassemble
118824	224.1430640	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=2960, ID=5682) [Reassemble
118825	224.1431860	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=4440, ID=5682) [Reassemble
118826	224.1432690	192.168.0.4	192.168.0.3	SIP	1060 Request: REGISTER sip:192.168.0.3 |
118827	224.1433940	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=5683) [Reassembled i
118828	224.1435230	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=5683) [Reassemble
118829	224.1436420	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=2960, ID=5683) [Reassemble
118830	224.1437650	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=4440, ID=5683) [Reassemble
118831	224.1438490	192.168.0.4	192.168.0.3	SIP	1060 Request: REGISTER sip:192.168.0.3 |
118832	224.1439750	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=5684) [Reassembled i
118833	224.1440980	192.168.0.4	192.168.0.3	IPv4	1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=5684) [Reassemble

Obrázek 5.23 Zachycené fragmentované pakety žádosti REGISTER generované aplikací SIPp.

Definici žádosti REGISTER zasílaná na telefonní ústřednu Asterisk byla rovněž uložena do souboru *register.xml* s tím rozdílem, že do sekce `<send> </send>` bylo přidáno několik set různých znaků za účelem zvětšení velikosti žádosti tak, aby byla fragmentovaná.

Nástroj SIPp se v tomto scénáři, stejně jako v předchozím, nesnaží sestavit reálné spojení. Zahlcuje síť opakujícími se fragmentovanými žádostmi REGISTER.

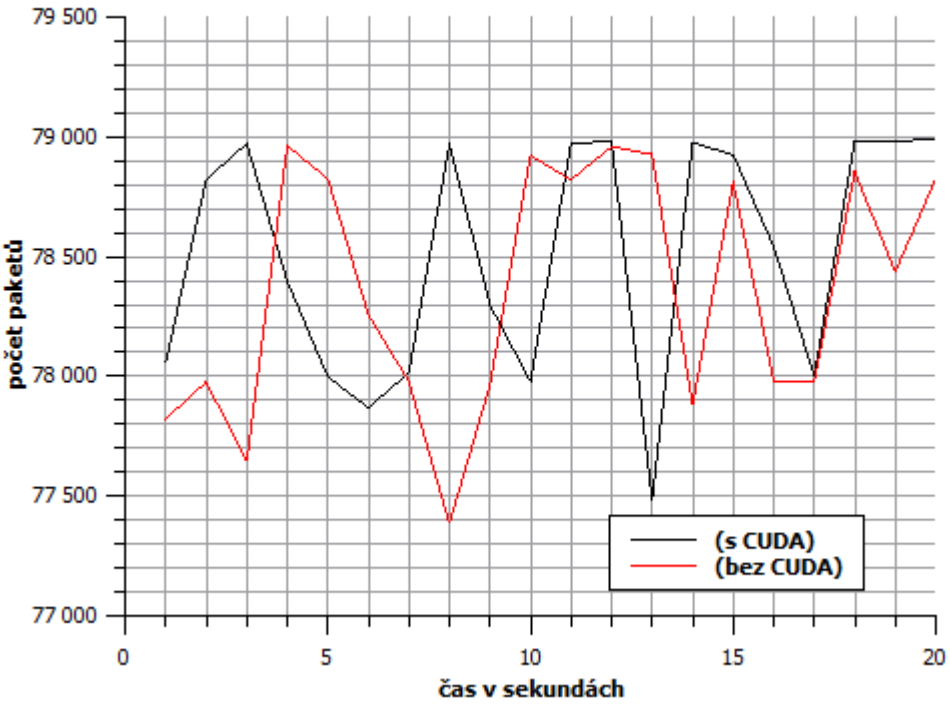
Při tomto testování bylo IDS Suricata s podporou technologie CUDA okamžitě detekováno 78.055 p/s a při dalším testování se výsledky nijak nelišily, viz tabulka 5.6 a obrázek 5.24. Při testování IDS bez podpory technologie CUDA byl okamžitý výsledek 77.822 p/s. Při testování, které bylo 20x opakováno, bylo zjištěno, že systémové nároky na IDS jsou v konfiguraci bez podpory technologie CUDA vyšší.

Využití CPU počítače při výkonovém testování IDS Suricata v konfiguraci s podporou technologie CUDA se pohybovalo v rozmezí 2,7-6,3% a při konfiguraci bez podpory technologie CUDA v rozmezí 13,2-20,9%.

Velikost využité paměti RAM IDS Suricata byl při konfiguraci s podporou technologie CUDA v rozmezí 1357-1411 MB a při konfiguraci bez podpory technologie CUDA v rozmezí 2240-2381 MB. Zde, při využití paměti RAM se projevilo, že IDS Suricata při konfiguraci s podporou technologie CUDA daleko lépe dokáže pracovat se systémovými prostředky hostitelského systému, kdy je IDS nucen detekované fragmentované pakety spojit a následně otestovat s nakonfigurovanými pravidly. Tato závislost na využití paměti RAM se tak výrazně neprojevila při žádném předešlém testování.

Tabulka 5.6 *Počet dekódovaných paketů nástroje SIPp generujícího fragmentovanou žádost REGISTER s podporou CUDA a bez podpory CUDA.*

	Počet paketů za sekundu	
sekundy	s CUDA	bez CUDA
1	78055	77822
2	78822	77976
3	78972	77645
4	78397	78963
5	77998	78826
6	77869	78258
7	78015	77978
8	78971	77385
9	78298	77964
10	77978	78923
11	78972	78821
12	78982	78959
13	77482	78926
14	78976	77882
15	78925	78811
16	78542	77982
17	78002	77976
18	78982	78854
19	78982	78436
20	78991	78827



Obrázek 5.24 *Grafické porovnání počtu dekódovaných paketů aplikace SIPp generující fragmentované žádosti REGISTER.*

Při tomto testování rovněž nedocházelo ke zpožděné detekci paketů. Při testování byl rozdíl ve využití paměti RAM o 883 MB vyšší při konfiguraci IDS Suricata bez podpory technologie CUDA a dále využití CPU o 10,5 – 14,6%. Příkon pracovního stroje s IDS/IPS Suricata byl v průběhu výkonnostního testování s konfigurací technologie CUDA 260W a bez konfigurace technologie CUDA 100W.

U tohoto měření byl nad daty získanými měřením proveden Shapiro-Wilkov test pro ověření normality dat na 95% hladině spolehlivosti, kterým bylo zjištěno, že data pochází z normálního rozdělení bez odlehlých pozorování nebo chyb. Poté byl proveden dvouhodnotový Studentův t-test, kterým bylo zjištěno, že naměřená data nejsou statisticky významně rozdílná na hladině pravděpodobnosti 90%, což znamená, že IDS Suricata detekuje množství paketů za sekundu stejně rychle v konfiguraci s podporou technologie CUDA i bez podpory CUDA, viz obrázek 5.25.

Sample	N	Mean	Standard Deviation	Variance	Standard Error
Table1_2	20	78 360,7	522,7166989569	273 232,7473684	116,8830071842
Table1_3	20	78 510,55	503,0185174995	253 027,6289474	112,478359907

Difference of Means: -149,85

Null Hypothesis: Mean1 - Mean2 = 0
Alternative Hypothesis: Mean1 - Mean2 <> 0

t	DoF	P Value
-0,9237860700332	38	0,3614267932582

At the 0,05 level, the difference of the population means is not significantly different than the test difference (0).

Obrázek 5.25 *Studentův test, zda jsou naměřená data statisticky významně rozdílná či nikoliv.*

5.5 Zhodnocení výsledků testů

Během testování bylo pohlíženo na několik parametrů a všechny provedené výkonové testy se od sebe nějakým způsobem lišily. Jednalo se zejména o různý počet portů přepínače, ze kterých byl generován provoz, různý počet generovaných paketů, jejich velikost a případná fragmentace. Tyto naměřené hodnoty byly přehledně zpracovány do grafů a tabulek u provedených jednotlivých výkonových měření a zde v této části bude provedeno srovnání výsledků testů z jednotlivých měření ve vztahu k využití systémových prostředků IDS/IPS Suricata, které jsou pro přehlednost uvedeny v tabulce 5.7. Dále v tabulce 5.8 je uveden příkon pracovního stoje s IDS/IPS Suricata a zpoždění detekce paketů pro porovnání s konfigurací a bez konfigurace technologie CUDA.

Tabulka 5.7 *Výsledky testů využití systémových prostředků IDS Suricata.*

název penetračního nástroje	počet portů generujících síťový VoIP provoz	celkový počet generovaných paketů IP telefonie **[paketů/s]**	využití systémových prostředků při konfiguraci IDS Suricata			
			s CUDA		bez CUDA	
			CPU **[%]**	RAM **[MB]**	CPU **[%]**	RAM **[MB]**
InviteFlood	1	162.284	7,1-12,3	3733 (max.)	15,4-19,8	3733 (max.)
Svcrack	2	22.172	1,3-5,4	1563	10,5-16,1	1823
SIPp a INVITE	2	120.853	1,6-6,7	1863	12,5-24,2	2185
SIPp a BYE	2	50.630	2,4-6,3	1625	10,7-20,1	1983
SIPp a REGISTER	1	42.865	1,3-5,6	1568	11-18,6	1853
SIPp a fragmentovaný REGISTER	1	78.055	2,7-6,3	1357	13,2-20,9	2240

Tabulka 5.8 *Výsledky testů detekce paketů a spotřeby IDS Suricata v konfiguraci s podporou technologie CUDA a bez podpory CUDA technologie.*

název penetračního nástroje	konfigurace IDS Suricata					
	s CUDA			bez CUDA		
	počet detekovaných paktů po spuštění **[paket/s]**	potřebný čas k detekci paketů bez zpoždění **[s]**	příkon stroje **[W]**	počet detekovaných paktů po spuštění **[paket/s]**	potřebný čas k detekci paketů bez zpoždění **[s]**	příkon stroje **[W]**
InviteFlood	95456	20	280	62329	87	120
Svcrack	19749	< 1	260	22866	< 1	110
SIPp a INVITE	51235	7	270	67972	4	120
SIPp a BYE	50628	0	250	50630	0	90
SIPp a REGISTER	42865	0	250	42325	0	70
SIPp a fragmentovaný REGISTER	78055	0	260	77822	0	100

Jak jde z tabulky 5.7 na první pohled vidět, je z dlouhodobého hlediska využití systémových prostředků IDS/IPS Suricata s konfigurací podpory technologie CUDA vyšší. Z detailnějšího rozboru výkonových testů uvedených v jednotlivých kapitolách musíme brát ale také ohled na zpoždění detekce paketů, zejména, že například detekce paketů při konfiguraci IDS Suricata s technologií CUDA u penetračního nástroje Svcrack je pomalejší a dochází zde ke zpoždění oproti konfigurace IDS Suricata bez technologie CUDA, což je ale vykoupeno vyššími nároky na systémové prostředky, tak jako u ostatních výkonových testů.

Z tabulky 5.7 lze dále vidět, že využití systémových prostředků je zejména závislé na tom, zda je či není IDS Suricata konfigurovaná s podporou technologie CUDA. Dále využití těchto systémových prostředků je závislé na tom, z kolika portů přepínače je provoz detekován a dále, jak by se dalo předpokládat na celkovém datovém objemu v reálném čase uvnitř testovací topologie.

Rozdíl využití paměti RAM se nejvíce projevila při výkonovém testování fragmentovaných paketů žádosti REGISTER a rozdíl využití CPU hostitelského sytému IDS Suricata se nejvíce projevil při detekci provozu ze dvou portů u výkonového testování nástrojem SIPp a zasíláním žádosti INVITE na kterou reagovala PBX Asterisk odpověďmi.

Pro zajímavost byl při jednotlivých testech zjišťován příkon stroje s nainstalovaným IDS Suricata v konfiguraci s podporou a bez podpory technologie CUDA komerčním měřičem příkonu BaseTech Cost Control 3000 CZ s max. odchylkou naměřené hodnoty 5% [57]. Tímto měřením bylo zjištěno, že se příkon počítače bez konfigurace IDS Suricata s technologií CUDA pohyboval v rozmezí 70-120Wh a příkon počítače při výkonovém testování s konfigurací IDS Suricata s podporou technologie CUDA

v rozmezí 245-280Wh, viz tabulka 5.8. Myslím si, že i s tímto měřeným parametrem je třeba při rozhodování se, v jakém prostředí bude IDS Suricata nasazena počítat. V době dokončení diplomové práce (duben 2016) se náklady na pořízení počítačové sestavy bez podpory technologie CUDA pohybovaly okolo 18.300 Kč a s podporou technologie CUDA okolo 35.700 Kč, viz tabulka 5.9. Ceny uvedené v tabulce 5.9 byly nezávisle na sobě získány z několika různých e-shopů v době dokončení diplomové práce (duben 2016).

Tabulka 5.9 *Porovnání cenových nákladů na počítačovou sestavu s podporou technologie CUDA a bez podpory technologie CUDA.*

	bez podpory CUDA (Kč)	S podporou CUDA (Kč)
Procesor i7-4770k	10.300	10.300
4 GB DDR3 RAM Kingston	500	500
Samsung 500 GB HDD	4.200	4.200
Základní deska	2.000	9.000
Zdroj	800	4.000
Skříň	500	3.000
nVidia GeForce GTX 660	-	4.700

Závěrem můžu tedy říct, že s ohledem na:

- pořizovací náklady počítačové sestavy podporující technologii CUDA,
- energetickou náročnost,
- testy významnosti rozdílů naměřených hodnot, které IDS Suricata detekoval v konfiguraci s podporou technologie CUDA a bez podpory technologie CUDA,
- testovací topologii IP telefonie, ve které byla propustnost 1Gbit/s,
- rezervu systémových prostředků počítačové sestavy nepodporující technologii CUDA,
- náročnost na propustnost sítě jednoho telefonního hovoru přenášeného RTP pakety (okolo 50Kbit/s),

bych raději volil počítačovou sestavu neumožňující podporu technologie CUDA. Dále také s ohledem na silnou podporu ze strany vývojářů IDS/IPS Suricata, kdy jsou hlavně pravidla průběžně aktualizovaná a doplňována, mimo jiné také za účelem snížení hrozeb útoků snažících se vyčerpat systémové prostředky PBX. Věřím však, že technologie CUDA má své místo v IP telefonii na topologii sítě přesahující 10Gbit/s.

6 Závěr

V diplomové práci jsem se snažil zmapovat penetrační nástroje testující zabezpečení IP telefonie v paketové síti a dále z těchto vybrat vhodné kandidáty k výkonovému testování IDS/IPS Suricata, která byla testována v konfiguraci s podporou technologie CUDA a bez ní.

Po úvodním seznámením se ze základy způsobu VoIP SIP komunikace a technologií CUDA v kapitole 1 byl v následující kapitole uveden Intrusivní detekční systém a jeho možnosti nasazení v síťové topologii. V této kapitole byl také představen IDS/IPS Suricata.

V následující kapitole byl uveden přehled penetračních nástrojů pro IP telefonii, kdy z těchto byly z povahy této diplomové práce v praktické části použity k testování penetrační nástroje, které aktivně zasahují do síťového provozu útokem DoS nebo slovníkovým útokem. Dále zde byl představen penetrační nástroj SIPp, který uměle vytváří SIP hovory za účelem otestování výkonnosti telefonní ústředny, v našem případě Asterisk, který byl s výhodou využit k výkonovému testování IDS/IPS Suricata.

Ve čtvrté kapitole je představena testovací topologie, která byla navržena v síti s podporou IP telefonie. Jednotlivými komponentami této topologie je pobočková ústředna Asterisk, zapojena do přepínače TP-LINK, který byl nakonfigurován k zrcadlení portů na port IDS/IPS Suricata. Do přepínače je dále zapojen hardwarový IP telefon zn. GRANDSTREAM a PC s operačním systémem Ubuntu 14.04 na kterém je nainstalován SIP softwarový klient YateClient. Do přepínače je dále zapojeno PC s operačním systémem Kali-linux, který představuje útočníka s komplexní sadou nástrojů k testování zabezpečení IP telefonie, v našem případě k výkonovému testování IDS/IPS Suricata.

Na základě výsledků testování různých penetračních nástrojů bylo zhodnoceno, zda v konkrétním případě je vhodné konfigurovat IDS/IPS Suricata s podporou technologie CUDA nebo bez ní. Výkonové testování bylo zaměřeno na využití hlavních systémových prostředků v reálném čase a to CPU a paměť RAM a dále na rychlost odezvy a rychlost získání stabilního stavu při detekci paketů nebo zpoždění těchto paketů detekovaných v síti IDS/IPS Suricata při generování síťového provozu nebo útoku.

Tato diplomová práce by měla být čtenáři vodítkem při vlastním rozhodování se, zda je pro jeho daný konkrétní případ, s ohledem na velikost počítačové sítě a její zatížení, ve které bude IDS/IPS Suricata nasazen vhodné investovat do technologie CUDA.

Literatura

[1] Nicholas, Wilt. The CUDA Handbook. 1. vydání United States, 2013, 487s. ISBN-13: 978-0-321-80946-9

[2] stažení CUDA 7 [Online] [Citace 23.5.2015] Dostupné z URL: https://developer.nvidia.com/cuda-downloads

[3] GeForce [Online] [Citace 23.5.2015] Dostupné z URL: http://www.nvidia.com/gtx-700-graphics-cards/gtx-titan-z/

[4] Victor, Malyshkin. Prallel Computing Technologies. 1. vydání Novosibirsk Rusia, 2015, 546s. ISBN 978-3-319-21908-0

[5] srovnání CPU a GPU [Online] [Citace 24.5.2015] Dostupné z URL: http://www.nvidia.com/object/what-is-gpu-computing.html

[6] architektura GPU [Online] [Citace 24.5.2015] Dostupné z URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/#axzz3b5UxqAfz

[7] Power 8-IBM technologie [Online] [Citace 3.2.2016] Dostupné z URL: https://www.ibm.com/developerworks/community/wikis/form/anonymous/api/wiki/61ad9cf2-c6a3-4d2c-b779-61ff0266d32a/page/f1abe75a-a2b2-43dd-9d75-7dae28f5bc5f/attachment/3d574a4b-b414-42c8-85b0-f941115d569f/media/2014-06%20Power%208%20Servers%20June.pdf.

[8] architektura GeForce GTX Titan Z [Onlline] [Citace 25.1.2016] Dostupné z URL: http://www.guru3d.com/articles-pages/nvidia-geforce-gtx-titan-z-review,6.html

[9] rozdíl architektur GPU a CPU [Online] [Citace 27.1.2016] Dostupné z URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#axzz3z91PAsVI

[10] rozdíl SIMD a SIMT [Online] [Citace 15.10.2015] Dostupné z URL: http://www.prednasky.com/lecture/playback/?lectureId=266

[11] AL, Sakib. Khan, Pathan. The State of the Art in Intrusion Prevention and Detection. 1. vydání, New York: CRC Press, 2014, 496 s. ISBN 978-1482203516

[12] Chris Sanders, Jason Smith: Applied Network Security Monitoring: Collection, Detection, and Analysis, 2013, ISBN-13: 978-0124172081

[13] Jason Sanders. CUDA by Example: An Introduction to General-Purpose GPU Programming, 2010, ISBN-13: 007-6092047179

[14] TP-LINK TL-SG105E [Online] [Citace 4.9.2015] Dostupné z URL: http://www.tp-link.com/en/products/details/cat-41_TL-SG105E.html

[15] GRANDSTREAM GXV3140 [Online] [Citace 2.1.2016] Dostupné z URL: http://www.grandstream.com/sites/default/files/Resources/gxp21xx_14xx_usermanual_english.pdf

[16] J. Postel. RFC 768 – user datagram protocol. Technical report, IETF, 1980.

[17] H.Schulzrine, S. Casner, R. Frederick, V. Jacobson. RFC 1889 – a transport protocol for real-time pplications. Technical report, IETF, 1996.

[18] H.Schulzrine, S. Casner, R. Frederick, V. Jacobson. RFC 3550 – a transport protocol for real-time pplications. Technical report, IETF, 2003.

[19] Radek Mandrla - SIP softwarový klient s podporou šifrování v prostředí Android, Bakalářská práce, 2013.

[20] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. RFC 3261 – sip: Session initiation protocol. Technical report, IETF, 2002.

[21] R. Sparks. RFC 3515 – The Session Initiation Protocol (SIP) Refer Method, IETF, 2003. Dokument dostupný na URL: http://www.ietf.org/rfc/rfc3515.txt.

[22] A. B. Roach. RFC 3265 – Session Initiation Protocol (SIP) – Specific Event Notification, IETF, 2002. Doument dostupný na URL: http://www.ietf.org/rfc/rfc3265.txt.

[23] K. Morneault, R. Dantu, G. Sidebottom, B. Bidulock, J. Heitz RFC 3331 – Signaling System 7 (SS7) Message Transfer Part 2 (MTP2): User Adaptation Layer, IETF, 2002. Dokument dostupný z URL: http://tools.ietf.org/rfc/rfc3331.txt.

[24] S. Donovan. RFC 2976 – The SIP INFO Method, IETF, 2000. Dokument dostupný z URL: http://www.ietf.org/rfc/rfc2976.txt.

[25] J. Rosenberg, H. Schulzrine. – Reliability of Provisional Responses in the Session Initiationn Protocol (SIP), IETF, 2002. Dokument dostupný z URL: http://www.ietf.org/rfc/rfc3262.txt

[26] B. Cambell, J. Rosenberg, H. Schulzrine, C. Huitema, D. Gurle. – Session Initiation Protocol (SIP) Extension for Instant Messaging, IETF, 2002. Dokument dostupný z URL: http://www.ietf.org/rfc/rfc3428.txt.

[27] SIP. CESNET. IPTelWiki: SIP [Online]. [Citace 8.1.2013]. Dostupné z URL: https://sip.cesnet.cz/cs/protokoly/sip.

[28] Miroslav Vozňák. –TECHNICKÉ PRINCIPY IP TELEFONIE. Teorie a praxe IP telefonie [Online]. 2004 [Citace 18.12.2012]. Dokument dostupný z URL: http://www.ip-telefon.cz/archiv/dok_osta/ipt-2004_Principy_IPtel.pdf.

[29] Miroslav Vozňák. – Voice over IP. 1. vyd. Ostrava: VŠB-TU Ostrava, 2008. 176 p. ISBN 978-80-248-1828-3.

[30] M. Handley, V. Jacobson, C. Perkins. – SDP: Session Description Protocol, IETF, 2006. Dokument dostupný z URL: http://www.ietf.org/rfc/rfc4566.txt

[31] Technická specifikace grafické karty NVIDIA GeForce 590 [Online] [Citace 8.7.2015]. Dostupné z URL: http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-590/specifications

[32] Technická specifikace procesoru i7-4770k [Online] [Citace 8.7.2015]. Dostupné z URL: http://ark.intel.com/products/75123/Intel-Core-i7-4770K-Processor-8M-Cache-up-to-3_90-GHz

[33] SLI technologie NVIDIA [Online] [Citace 9.7.2015]. Dostupné z URL: http://nvidia.czc.cz/sli.html

[34] Suricata.yaml [Online] [Citace 18.1.2016]. Dostupné z URL: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricatayaml

[35] Ettercap [Online] [Citace 20.1.2016] Dostupné z URL: https://ettercap.github.io/ettercap/index.html

[36] InviteFlood [Online] [Citace 20.1.2016] Dostupné z URL: http://tools.kali.org/sniffingspoofing/inviteFlood

[37] nmap [Online] [Citace 21.1.2016] Dostupné z URL: https://nmap.org/

[38] Ec, Council. Penetrration Testing Communication Media Testing. 1, vydání New York, 2011, ISBN-13: 978-1-4354-8369-9

[39] SIPvicious [Online] [Citace 22.1.2016] Dostupné z URL: http://tools.kali.org/sniffingspoofing/sipvicious

[40] VoIPhopper [Online] [Citace 23.1.2016] Dostupné z URL: http://voiphopper.sourceforge.net/details.html

[41] OhrWurm [Online] [Citace 24.1.2016] Dostupné z URL: http://tools.kali.org/vulnerability-analysis/ohrwurm

[42] OpenVAS [Online] [Citace 24.1.2016] Dostupné z URL: http://openvas.org/software.html

[43] THC-Hydra [Online] [Citace 26.1.2016] Dostupné z URL: http://tools.kali.org/password-attacks/hydra

[44] THC-Hydra [Online] [Citace 27.1.2016] Dostupné z URL: https://www.thc.org/thc-hydra/

[45] rtpbreak [Online] [Citace 25.1.2016] Dostupné z URL: http://tools.kali.org/sniffingspoofing/rtpbreak

[46] rtpflood [Online] [Citace 25.1.2016] Dostupné z URL: http://tools.kali.org/stress-testing/rtpflood

[47] rtpinsertaudio [Online] [Citace 25.1.2016] Dostupné z URL: http://tools.kali.org/sniffingspoofing/rtpinsertsound

[48] rtpmixsound [Online] [Citace 25.1.2016] Dostupné z URL: http://tools.kali.org/sniffingspoofing/rtpmixsound

[49] SipArmyKnife [Online] [Citace 25.1.2016] Dostupné z URL: http://tools.kali.org/vulnerability-analysis/siparmyknife

[50] Xplico [Online] [Citace 26.1.2016] Dostupné z URL: http://tools.kali.org/information-gathering/xplico

[51] Xplico [Online] [Citace 26.1.2016] Dostupné z URL: http://www.xplico.org/

[52] Ace-VoIP [Online] [Citace 27.1.2016] Dostupné z URL: http://ucsniff.sourceforge.net/usage.html

[53] John-the-ripper [Onlina] [Citace 28.1:2016] Dostupné z URL: http://tools.kali.org/?s=john+the+ripper

[54] EnumIAX [Online] [Citace 28.1.2016] Dostupné z URL: http://tools.kali.org/information-gathering/enumiax

[55] Suricata.yaml [Online] [Citace 29.1.2016] Dostupné z URL: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricatayaml

[56] Syntaxe příkazu crunch [Online] [Citace 28.2.2016] Dostupné z URL: http://manpages.ubuntu.com/manpages/utopic/man1/crunch.1.html

[57] Technické parametry měřiče příkonu [Online] [Citace 3.3.2016] Dostupné z URL: http://www.conrad.com/ce/en/product/125333/Basetech-COST-CONTROL-3000-LCD-000-999999-kWh-

Seznam příloh

Příloha A: konfigurační soubor `suricata.yaml`

Tento soubor je základním konfiguračním souborem IDS/IPS Suricata. Lze zde nastavit konkrétní chování Suricaty (IDS nebo IPS), jak velkou část systémových prostředků může Suricata využít v různých nastaveních, využití technologie CUDA a další [55].

Nastavení chování jako IDS/IPS/AUTO:

```
host-mode: auto
```

Nastavení výchozího adresáře .log souborů:

```
default-log-dir: /var/log/suricata/
```

Dalším nastavením lze nakonfigurovat, kolik paketů může Suricata maximálně souběžně analyzovat. Nastavit lze od jednoho paketu až po stovky tisíc paketů. S výkonností však roste potřebná velikost paměti RAM [55].

```
max-pending-packets: 1024
```

Dále lze nastavit maximální velikost paketu, který Suricata může analyzovat. To záleží na konfiguraci sítě ve které je IDS nasazován. Je třeba ale uvažovat, že se výjimečně může přenášet i větší paket, což se však odrazí na výkonu IDS [55]. Nastavení lze provést:

```
default-packet-size: 1514
```

Lze nastavit, co se má s paketem, který vyhovuje některému z pravidel, provést.

- **PASS** - Suricata zastaví skenování a skočí pro daný paket na konec všech pravidel (paket je ignorován).

- **DROP** - pokud vyhovuje paket pravidlům, tak paket okamžitě zahodí a pakety nebudou dále zpracovávány. Suricata vygeneruje ALERT pro tento paket.

- **REJECT** - paket bude v případě, že vyhovuje pravidlům zahozen a odesílatel i příjemce v případě TCP protokolu dostane zprávu, že paket porušil pravidla pro TCP spojení (TCP reset). Dojde k zavření spojení .Pro protokol UDP bude paket brán jako ICMP-error (port bude pro odesílatele nedostupný). Pokud je Suricata nastavená jako IDS, bude s paketem zacházeno jako v případě nastavení DROP.

- **ALERT** - bude vygenerováno varování v log souboru a nic víc se s paketem nebude dělat.

Pořadí, v jakém se mají pravidla vykonávat lze nastavit následně. Defaultně je nastaveno: PASS, DROP, REJECT, ALERT. Znamená to, že paket, který vyhovuje všem pravidlům je nejdříve propuštěn dále a až pak zahozen, dále nejdříve je zahozen a až pak je zaslána odpověď (v případě TCP), že paket je odmítnut přijmout a nakonec je vygenerováno varování [55].

```
action-order:
- pass
- drop
```

```
- reject
- alert
```

Nastavení `http-log` obsahuje http žádosti, adresu hosta, URI a jméno uživatele. Tyto informace budou uloženy v `hhtp.log` [55].

```
http-log:
      enable: yes
      filename: http.log
      append: yes/no # jestliže nastavíme "yes", tak obsah souboru
                       nebude při příštím restartu Suricaty přepsán.
      Extended: yes  # pokud nastavíme "yes" bude zaznamenáno více
                       Informací o výjimce.
```

Nastavení `dns-log` obsahuje dns žádosti a odpovědi. Zaznamenávají se data, jako klient, server, ttl [55].

```
dns-log:
      enabled: yes
      filename: dns.log
      append: yes
```

Nastavením `pcap-log` můžeme ukládat všechny pakety, které Suricata zaznamenala do logu s názvem `log.pcap`. Tímto jsou zaznamenané pakety kdykoliv k nahlédnutí. Soubor může být otevřen jakýmkoliv programem, který podporuje formát pcap, například Wireshark. Pcap-log může být zapnutý nebo vypnutý. Výchozí limit velikosti souboru je 32 MB. Po dosažení této velikosti se začnou data přepisovat, tzv. rotovat [55].

```
pcap-log:
      enabled: yes
      filename: log.pcap
      limit: 32  # max. velikost souboru
```

V souboru `alert-debug.log` lze nastavit doplňkové informace o varování [55].

```
alert-debug:
      enabled: no
      filename: alert-debug.log
      append: yes/no
```

Nastavení ukládání informací obsažených ve statistikách souboru `stats.log` lze nastavit v sekundách, kdy mají být výstupní data ukládána do souboru [55].

```
- stats:
  enabled: yes
  filename: stats.log
  interval: 8
  append: yes/no
```

Nastavením `syslog` je možné zasílat varování do syslogu [55]

```
- syslog:
  enabled: no
  facility: local5      # lze nastavit konkrétní výstupní zařízení
  level: Info           # lze nastavit, která hlášení se mají
                         v syslogu zaznamenávat (Emergency, Alert,
                         Critical, Error, Warning, Notice, Info,
                         Debug).
  append: yes/no
```

V souboru `drop.log` lze nastavit ukládání zahozených paketů vyhovující podmínce pro zahození [55].

```
- drop:
  enabled: yes
  filename: drop.log
  append: yes/no
```

V části `detect-engine` jsou vytvářeny vnitřní skupiny signatur, které Suricata využívá, jelikož porovnávání velkého množství pravidel by mělo značný dopad na výkon. Například v síti, kde se často neobjevuje TCP protokol, ale protokol UDP by porovnávání pravidel obsahující protokol TCP bylo neefektivní. Proto jsou signatury rozděleny do skupin, které jsou uspořádány do stromové struktury. Každá signatura však není zařazena do určité skupiny, jelikož by toto měla dopad na zatížení paměti zařízení. Parametrem `profile` lze rozdělení do skupin nastavit. To znamená, menší zatížení paměti je vykoupeno menším výkonem a naopak vyšší zatížení paměti zvýší výkon IDS [55].

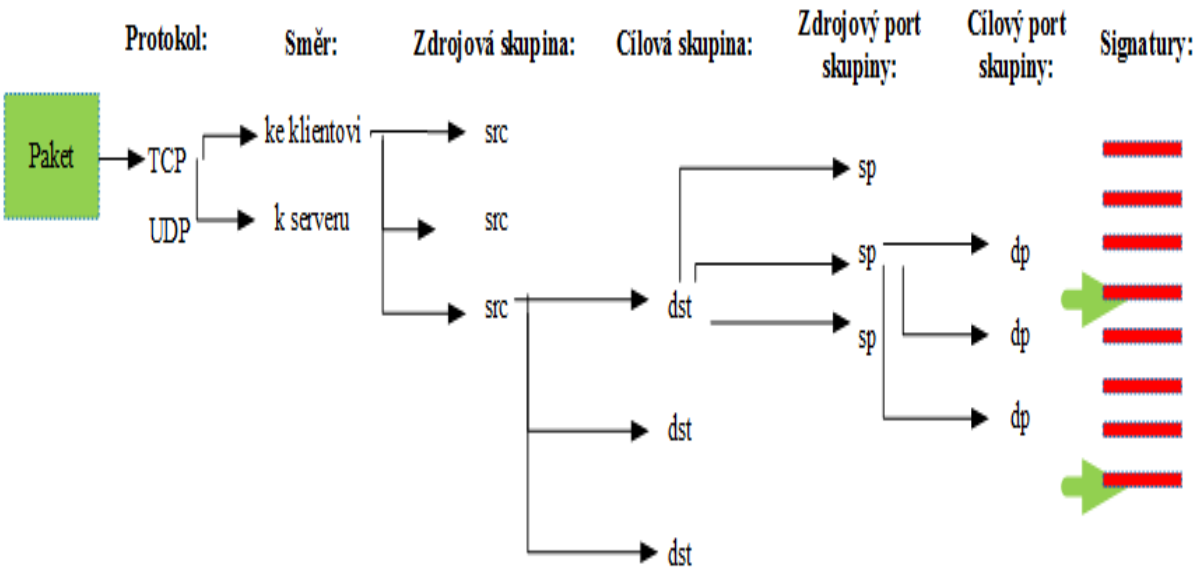
V parametru `profile` lze nastavit **`high`** pro vysoký výkon a vysoké zatížení paměti, **`low`** pro nízký výkon a malé zatížení paměti a **`medium`** které balancuje mezi výkonem a zatížením paměti zařízení s IDS/IPS Suricata. Nastavením `custom-values` lze nastavit kolik skupin má být vytvořeno pro klienta a server a u každého lze rozdělit skupiny podle adresy zdroje, adresy cíle, zdrojového portu a cílového portu [55].

Parametrem `Sgh-mpm-context` lze nastavit podle jakého vzoru budou jednotlivé skupiny prohledávány za účelem nalezení shody s pravidlem. Je navrženo několik algoritmů prohledávání skupiny, ale některé potřebují ke své činnosti větší paměť. Pro menší využití paměti je vhodné zvolit sdílený vzor prohledávání pro všechny skupiny [55]. Lze zvolit z možností **`auto`**, **`full`** nebo

single. Volba `full` znamená, že každá skupina má svůj vlastní vzor vyhledávání a volba `single` znamená, že všechny skupiny sdílejí stejný vzor vyhledávání pravidla. Pokud by došlo při prohledávání skupiny podle daného vzoru k zacyklení, lze nastavit max. počet opakování vyhledávání v položce `inspection-recursion-limit`. Na obrázku je názorně ukázáno přiřazení paketu dané skupině ve stromové struktuře.

```
detect-engine:

   - profile: medium
   - custom-values:
       toclient_src_groups: 2    # ke klientovi, zdrojová adresa
                                   skupiny
       toclient_dst_groups: 2    # ke klientovi, cílová adresa skupiny
       toclient_sp_groups: 2     # ke klientovi, zdrojový port skupiny
       toclient_dp_groups: 3     # ke klientovi, cílový port skupiny
       toserver_src_groups: 2
       toserver_dst_groups: 4
       toserver_sp_groups: 2
       toserver_dp_groups: 25
   - sgh-mpm-context: auto
   - inspection-recursion-limit: 3000
```



Obrázek 6.1 *Přiřazení paketu do skupiny ve stromové struktuře.*

Jak bylo uvedeno výše, Suricata také podporuje CUDA (Compute United Device Architecture). Tímto se potřebný výkon Suricaty přesouvá z CPU na GPU NVIDIA. Pomocí CUDA dokáže

zpracovávat více vzorové vyhledávání (multi-pattern-matcher). Můžeme nastavit několik voleb. Volbou `packet_buffer_limit` se nastavuje jaké množství paketů bude zasláno na GPU ve stejnou dobu [55].

Suricata odesílá pakety v sériích, to znamená, že vyšle více paketů současně. Jakmile je shromážděno množství paketů uvedené ve volbě `buffer_limit_option` odešle je GPU. Výchozí počet je 2.400 [55].

Nastavením `packet_size_limit` se omezuje velikost paketu. Tedy pakety, které přesáhnou nastavenou hodnotu (výchozí hodnota je 1.500) nebudou zasílány na GPU [55].

Nastavením `packet_buffers` se nastavuje velikost vyrovnávací paměti, která bude po naplnění zpracována. Ve vyrovnávací paměti se shromažďují dávky paketů. Po odeslání se začne vyrovnávací paměť opět plnit [55].

Nastavením `batching_timeout` se nastavuje perioda ve které budou pakety poslány na GPU ať už je vyrovnávací paměť zaplněna či nikoliv [55].

Nastavením `page_locked` lze nastavit, zda bude paměť uzamčena či nikoliv. Tím se myslí, že pokud bude paměť uzamčena, tak Suricata nemůže odkládat data na disk, což značně zpomaluje běh IDS/IPS [55].

Nastavením `device_id` lze nastavit v případě, že máme na zařízení připojeno více grafických karet podporujících technologii CUDA, kterou z GPU má Suricata pro svou činnost používat [55]. Id jednotlivých karet získáme po zadání příkazu `suricata --list-cuda-cards`.

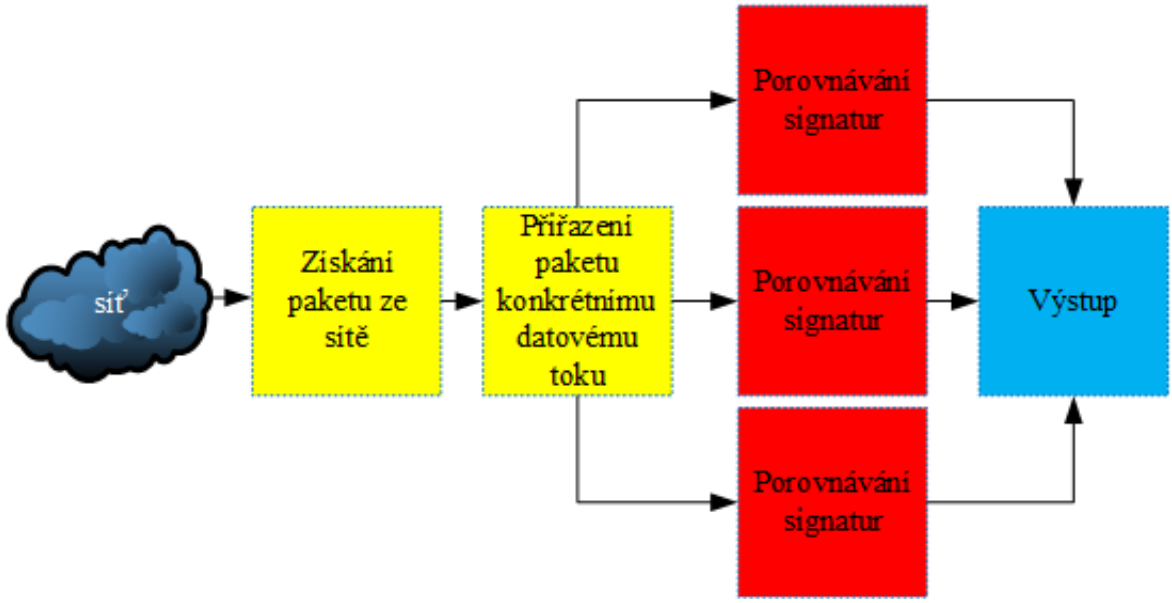
```
cuda:
  -mpm:
     packet_buffer_limit: 2400
     packet_size_limit: 1500
     packet_buffers: 10
     batching_timeout: 1
     page_locked: enabled
     device_id: 0
     cuda_streams: 2
```

Nastavením `multi-pattern-matcher` (MPM) můžeme měnit vyhledávací vzory u různých signatur. Suricata dokáže vyhledávat podle různých vyhledávacích vzorů najednou. Obecně platí, že signatura odpovídá jednomu nebo více vyhledávacím vzorům. Každá signatura je tedy obsažena v jednom vzoru používaném `multi-pattern-matcher`. Suricata tímto způsobem může pomocí různých vzorů určit přesný typ signatury, pokud odpovídá více (multi) vyhledávacím vzorům (pattern matcher) [55]. To znamená, že příchozí paket je analyzován pomocí `multi-pattern-matcher` a všechny záznamy, které odpovídají vyhledávacím vzorům, budou dále zasány IDS/IPS Suricata. Na výběr je nabízeno několik vyhledávacích algoritmů: `b2g, b2gc, b2gm, b3g, wumanber, ac and ac-gfbs`.

Surikata je více-vláknová, tím se myslí, že využívá jádra procesorů pro jednotlivá vlákna, což umožňuje zpracovávat velké množství síťových paketů současně. Pokud bude využito pouze jednoho jádra procesoru, budou pakety zpracovávány postupně. Surikata obsahuje čtyři vláknové moduly (obr. 6-2). Jedná se o získávání paketů, dekódování toku na aplikační vrstvě, detekce a výstup. Modul získávání paketů čte pakety ze sítě. Modul dekódování toku na aplikační vrstvě má tři úkoly:

- podniknuty veškeré kroky k získání kompletního toku dat,
- TCP síťový provoz bude kompletně rekonstruován,
- Aplikační vrstva bude kontrolována. HTTP a DCE/RPC bude analyzován.

Detekční vlákna budou porovnávat signatury. Může být několik detekčních vláken, které mohou pracovat současně. Na výstupu budou zpracovány všechna varování a výjimky.



Obrázek 6.2 *Více-vláknová podpora.*

```
Packet acquisition: čte pakety ze sítě
Decode:             dekóduje pakety
Stream app. Layer:  provádí zjišťování toku dat a jeho rekonstrukci
Detect:             porovnává signatury
Outputs:            zpracovává veškeré vyjímky a varování.
```

Nastavením `detect-thread-ratio: 1.5` nastavíme násobek detekčních vláken. Přednastavená hodnota je 1,5 násobek jader CPU zařízení. Ve výsledku vznikne více detekčních vláken, než je počet jader. Toto převýšení množství vláken oproti počtu jader procesorů se využívá proto, aby se zamezilo čekání na detekční vlákno.

Volbou `cpu-affinity` lze nastavit které jádro bude pracovat na kterém vlákně. Lze také nastavit kolik jader procesoru bude pracovat na daném vlákně (`management, receive, decode, stream, detect, verdict, reject a outputs`) [55]. U každého vlákna lze nastavit jeho priorita, která může být `low, medium, high` nebo `default.` Pokud priorita nebude nastavena bude se brát výchozí (default) hodnota. Ukázka nastavení `cpu-affinity` je uvedena níže.

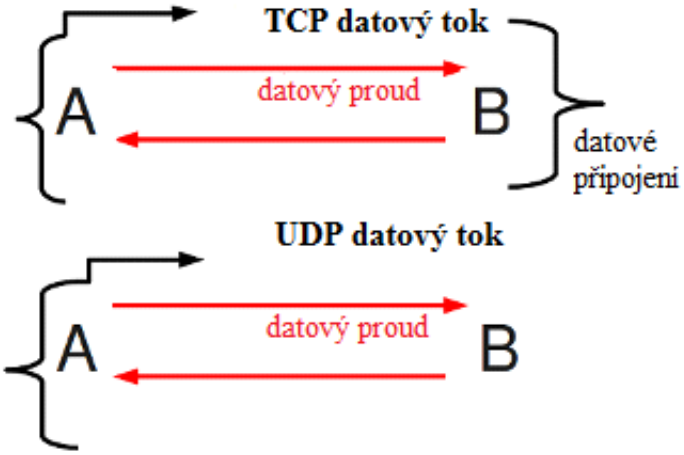
```
cpu-affinity:
    - management-cpu-set:
        cpu: [ 0 ]
    - receive-cpu-set:
        cpu: [ 0 ]
    - decode-cpu-set:
        cpu: [ 0, 1 ]
        mode: "balanced"
    - stream-cpu-set:
        cpu: [ "0-1" ]
    - detect-cpu-set:
        cpu: [ "all" ]
        mode: "exclusive"
        prio:
          low: [ 0 ]
          medium: [ "1-2" ]
          high: [ 3 ]
          default: "medium"
    - verdict-cpu-set:
        cpu: [ 0 ]
        prio:
          default: "high"
    - reject-cpu-set:
        cpu: [ 0 ]
        prio:
          default: "low"
    - output-cpu-set:
        cpu: [ "all" ]
        prio:
           default: "medium"
```

Další nastavení řeší fragmentování paketů, které se přenáší počítačovou sítí. Fragmentované pakety se skládají z mnoha částí. Pokud má být Suricata schopná analyzovat paket, musí jej přesně rekonstruovat. V části `defraf` lze nastavit tři části, kterýmidefinujeme, jak se má Surikata zachovat v případě přijetí fragmentovaného paketu. `max-frag, prealloc` a `timeout`. V případě přijetí fragmentovaného paketu si jej Surikata drží v paměti, dokud nedorazí všechny části fragmentovaného paketu. V případě, že nedorazí všechny fragmenty paketu, drží si přijaté fragmenty v paměti po dobu 60 sekund a pak je zahodí, aby nealokovali paměť.

```
defrag:
  max-frags: 65535
  prealloc: yes
```

Dále lze upravit nastavení datového toku, který si Surikata umí vyhodnotit a zpracovávané pakety přiřazuje danému datovému toku. Do stejného datového toku Surikata přiřazuje pakety, které

mají stejný přenosový prokokol, stejnou zdrojovou a cílovou adresu, zdrojový a cílový port.



Obrázek 6.3 *Datový tok.*

Samozřejmě, že čím více datových toků si Surikata udržuje, tím více je zaplňována paměť. Aby nedošlo k přeplnění paměti, je toto ošetřeno v části `flow`. Nastavení této části je třeba věnovat pozornost i z důvodu, že by případný útočník mohl na výkonnost Surikaty zaútočit právě způsobem, že by vytvářel nové a nové datové toky, čímž by mohlo k přeplnění paměti dojít.

```
flow:
  memcap: 33554432  # maximální počet Byte všech datových
                    toků
  hash_size: 65536 # datové toky jsou uspořádány v hašovací
                    tabulce, kdy zde nastavujeme její max.
             velikost
  Prealloc: 10000   # množství datových toků udržovaných v
                    paměti
```

Pokud by i přes výše uvedené nastavení došlo k přeplnění paměti, přepne se Surikata do `emergency-mode`. V tomto módu se Surikata začne chovat více agresivně a začne ukončovat staré datové toky, aby uvolnila místo novým.

```
emergency_recovery: 30 # procento z 10000 alokovaných datových
                       toků
prune_flows: 5           # počet ukončených datových toků,
                pokud se Surikata přepne do emergency-mode
```

Dalším nastavením určujeme výchozí adresář, kde jsou uloženy soubory s pravidly Surikaty, která jsou rozdělena do souborů podle rizika nebezpečí útoku.

```
default-rule-path: /etc/suricata/rules/
rule-files:
  - backdoor.rules
  - bad-traffic.rules
```

```
  - chat.rules
  - ddos.rules
  - ....
```

Nastavení proměnných, které jsou použity v soborech s pravidly můžeme upravit v nastavení `vars`. Můžeme nastavit IP sdresu na které bude síťový provoz Surikatou kontrolován a na které ne. Nastavením adresy `HOME_NET` nastavíme IP adresu domácí sítě [55].

```
Vars:
  address-groups:
 HOME_NET: # například "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
 EXTERNAL_NET: any
 HTTP_SERVERS: "$HOME_NET" # znak $ znamená, že bude
                           následovat proměnná
 SMTP_SERVERS: "$HOME_NET"
 SQL_SERVERS: "$HOME_NET"
 DNS_SERVERS: "$HOME_NET"
 TELNET_SERVERS: "$HOME_NET"
 AIM_SERVERS: any
```

Existují dva typy proměnných, kdy jeden je ve skupině adres a druhý typ je ve skupině portů. Oba dva typy jsou nastavitelné z důvodu, aby byla pravidla smysluplná.

```
port-groups:
  HTTP_PORTS: "80"
  SHELLCODE_PORTS: "!80"
  ORACLE_PORTS: 1521
  SSH_PORTS: 22
```